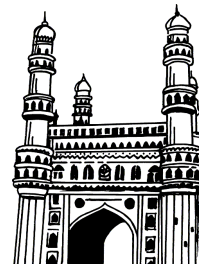


Rahul's ✓
Topper's Voice

**NEW
SYLLABUS**



M.Sc.

(COMPUTER SCIENCE)

II Year IV Sem

(Osmania University)

Latest Edition

INTERNET OF THINGS

☞ **Study Manual**

☞ **Solved Model Papers**

- by -

WELL EXPERIENCED LECTURER

Price
₹. 199-00



Rahul Publications™

Hyderabad. Ph : 66550071, 9391018098

All disputes are subjects to Hyderabad Jurisdiction only

M.Sc.

II Year IV Sem

INTERNET OF THINGS

Inspite of many efforts taken to present this book without errors, some errors might have crept in. Therefore we do not take any legal responsibility for such errors and omissions. However, if they are brought to our notice, they will be corrected in the next edition.

© No part of this publications should be reporduced, stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, photocopying, recording and/or otherwise without the prior written permission of the publisher

Sole Distributors :

Cell : 9391018098, 9505799122

VASU BOOK CENTRE

Shop No. 2, Beside Gokul Chat, Koti, Hyderabad.

Maternity Hospital Opp. Lane, Narayan Naik Complex, Koti, Hyderabad.

Near Andhra Bank, Subway, Sultan Bazar, Koti, Hyderabad -195.

INTERNET OF THINGS

CONTENTS

Unit - I	1 - 21
Unit - II	22 - 39
Unit - III	40 - 56
Unit - IV	57 - 96
Model Paper - I	97 - 97
Model Paper - II	98 - 98

SYLLABUS

UNIT - I

Introduction to Internet of Things : Introduction, Physical Design of IoT, Logical Design of IoT, IoT Enabling Technologies, IoT Levels & Deployment Templates, Domain Specific IoTs: Home Automation, Cities, Environment, Energy, Retail, Agriculture, Health & Lifestyle.

UNIT - II

IoT and M2M: Introduction to M2M, Difference between IoT and M2M, SDN and NFV for IoT. IoT System Management with NETCONF-YANG: Need for IoT Systems Management, SNMP, Network Operator requirements, NETCONF, YANG, IoT Systems Management with NETCONF-YANG. IoT Platforms Design Methodology: Introduction, IoT Design Methodology, Case Study on IoT system for weather Monitoring. Motivation for Using Python. Python Packages for IoT.

UNIT - III

IoT Physical Devices & Endpoints: What is an IoT Device, Exemplary Device: Raspberry Pi, About the Board, Linux on Raspberry Pi, Raspberry Pi Interfaces, programming Raspberry Pi with Python, Other IoT Devices.

IoT Physical Servers & Cloud Offerings: Introduction to Cloud Storage Models & Communication APIs, WAMP AutoBahn for IoT, Xively Cloud for IoT, Python Web Application Framework-Django, Designing a RESTful Web API, Amazon Web Services for IoT, SkyNet IoT Messaging Platform.

UNIT - IV

Case Studies of IoT Design: Home Automation, Cities, Environment, Agriculture, Productivity Applications. Introduction to Data Analytics for IoT, Apache Hadoop, YARN, Oozie, Spark, Storm, Health Monitoring Case study. An IoT Tool: chef, Chef Case Studies.

Contents

Topic No.		Page No.
UNIT - I		
1.1	Introduction to Internet of Things	1
1.1.2	Physical Design of Iot	2
1.1.3	Logical Design of IOT	7
1.1.4	IOT Enabling Technologies	11
1.1.5	IOT Levels & Deployment Templates	12
1.2	Domain Specific Iots	16
1.2.1	Home Automation	16
1.2.2	Cities	17
1.2.3	Environment	18
1.2.4	Energy	19
1.2.5	Retail	20
1.2.6	Agriculture	20
1.2.7	Health & Lifestyle	21
UNIT - II		
2.1	IOT and M2M	22
2.1.1	Introduction To M2M	22
2.1.2	Difference Between Iot And M2M	24
2.1.3	Sdn and Nfv for Iot	25
2.2	IOT System Management With Netconf-Yang	28
2.2.1	Need for Iot Systems Management	28
2.2.2	SNMP	28
2.2.3	Network Operator Requirements	29
2.2.4	NETCONF	30
2.2.5	Yang	31
2.2.6	IOT Systems Management w With Netconf-yang	32
2.3	IOT Platforms Design Methodology	33
2.3.1	Introduction	33
2.3.2	IOT Design Methodology	34
2.3.3	Case Study on Iot System for Weather Monitoring	35
2.3.4	Motivation for Using Python	37
2.3.5	Python Packages for Iot	37

Topic No.		Page No.
UNIT - III		
3.1	IOT Physical Devices & Endpoints	40
3.1.1	What is an IoT Device	40
3.2	Exemplary Device	41
3.2.1	Raspberry PI	41
3.2.2	About the Board	41
3.2.3	Linux on Raspberry Pi	43
3.2.4	Raspberry Pi Interfaces	45
3.2.5	Programming Raspberry Pi with Python	45
3.2.6	Other IoT Devices	47
3.3	IOT Physical Servers & Cloud Offerings	48
3.3.1	Introduction to Cloud Storage Models & Communication Apis	48
3.3.2	WAMP Autobahn for IoT	48
3.3.3	Xively Cloud for IoT	49
3.3.4	Python Web Application Framework-django	51
3.3.5	Designing a Restful Web Api	52
3.3.6	Amazon Web Services for IoT	54
3.3.7	Skynet IoT Messaging Platform	56
UNIT - IV		
4.1	Case Studies of IoT Design	57
4.1.1	Home Automationand Cities	57
4.1.2	Environment	62
4.1.3	Agriculture	65
4.1.4	Productivity Applications	69
4.1.5	Introduction to Data Analytics for IoT	70
4.1.6	Apache Hadoop	70
4.1.7	YARN	74
4.1.8	OOZIE	76
4.1.9	Spark	78
4.1.10	Storm	81
4.1.11	Health Monitoring Case Study	86
4.2	An IOT Tool	88
4.2.1	CHEF	88
4.2.2	Chef Case Studies	93

UNIT I

Introduction to Internet of Things : Introduction, Physical Design of IoT, Logical Design of IoT, IoT Enabling Technologies, IoT Levels & Deployment Templates, Domain Specific IoTs: Home Automation, Cities, Environment, Energy, Retail, Agriculture, Health & Lifestyle.

1.1 INTRODUCTION TO INTERNET OF THINGS

1.1.1 Introduction

Q1. What is IoT ? Explain about it.

Ans :

Internet of Things represents a general concept for the ability of network devices to sense and collect data from the world around us, and then share that data across the Internet where it can be processed and utilized for various interesting purposes.

Some also use the term industrial Internet interchangeably with IoT. This refers primarily to commercial applications of IoT technology in the world of manufacturing. The Internet of Things is not limited to industrial applications, however.

What the Internet of Things Can Do for Us

Some future consumer applications envisioned for IoT sound like science fiction, but some of the more practical and realistic sounding possibilities for the technology include :

- ▶ receiving warnings on your phone or wearable device when IoT networks detect some physical danger is detected nearby (like : smart smoke detectors)
- ▶ self-parking automobiles (like : Volvo S90)
- ▶ automatic ordering of groceries and other home supplies (like : Amazon Dash Wand)
- ▶ automatic tracking of exercise habits and other day-to-day personal activity including goal tracking and regular progress reports (like : fitness trackers)

Potential benefits of IoT in the business world include :

- ▶ location tracking for individual pieces of manufacturing inventory
- ▶ fuel savings from intelligent environmental modeling of gas-powered engines
- ▶ new and improved safety controls for people working in hazardous environments

How the Internet of Things Works

The Internet of Things (IoT), also sometimes referred to as the Internet of Everything (IoE), consists of all the web-enabled devices that collect, send and act on data they acquire from their surrounding environments using embedded sensors, processors and communication hardware. These devices, often called "connected" or "smart" devices, can sometimes talk to other related devices, a process called **machine-to-machine(M2M)** communication, and act on the information they get from one another.

Humans can interact with the gadgets to set them up, give them instructions or access the data, but the devices do most of the work on their own without human intervention. Their existence has been made possible by all the tiny mobile components that are available these days.

Connected devices also generate massive amounts of Internet traffic, including loads of data that can be used to make the devices useful, but can also be mined for other purposes. All this new data, and the Internet-accessible nature of the devices, raises both privacy and security concerns.

1.1.2 Physical Design of IOT

Q2. Explain about the layers of IoT architecture.

Ans :

The "Things" in IoT usually refers to IoT devices which have unique identities and can perform remote sensing, actuating and monitoring capabilities.

IoT devices can :

- ▶ Exchange data with other connected devices and applications (directly or indirectly), or
- ▶ Collect data from other devices and process the data locally or
- ▶ Send the data to centralized servers or cloud-based application back-ends for processing the data, or
- ▶ Perform some tasks locally and other tasks within the IoT infrastructure, based on temporal and space constraints

IoT Architecture Layers

There are four major layers.

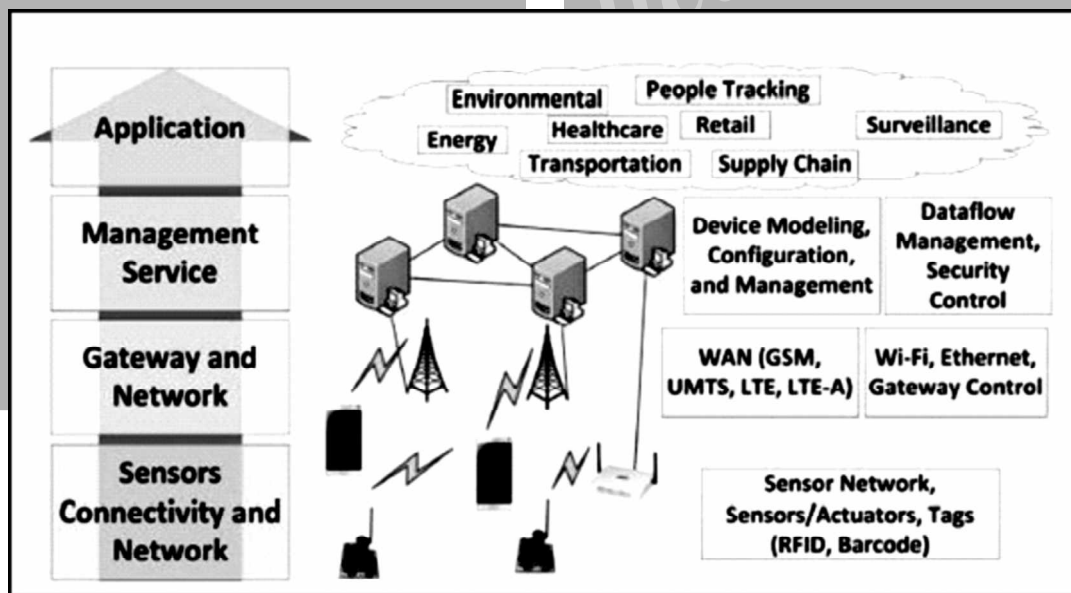


Fig. : IoT architecture layers

At the very bottom of IoT architecture, we start with the Sensors and Connectivity network which collects information. Then we have the Gateway and Network Layer. Above which we have the Management Service layer and then at the end we have the application layer where the data collected are processed according to the needs of various applications.

The features of each layer:

Sensor, Connectivity and Network Layer

This layer consists of RFID tags, sensors (which are essential part of an IoT system and are responsible for collecting raw data). These form the essential “things” of an IoT system.

- ▶ Sensors, RFID tags are wireless devices and form the Wireless Sensor Networks (WSN).
- ▶ Sensors are active in nature which means that real-time information is to be collected and processed.
- ▶ This layer also has the network connectivity (like WAN, PAN etc.) which is responsible for communicating the raw data to the next layer which is the Gateway and Network Layer.
- ▶ The devices which are comprised of WSN have finite storage capacity, restricted communication bandwidth and have small processing speed.
- ▶ We have different sensors for different applications – temperature sensor for collecting temperature data, water quality for examining water quality, moisture sensor for measuring moisture content of the atmosphere or soil etc.

As per the figure below, at the bottom of this layer we have the tags which are the RFID tags or barcode reader, above which we have the sensors/actuators and then the communication networks.

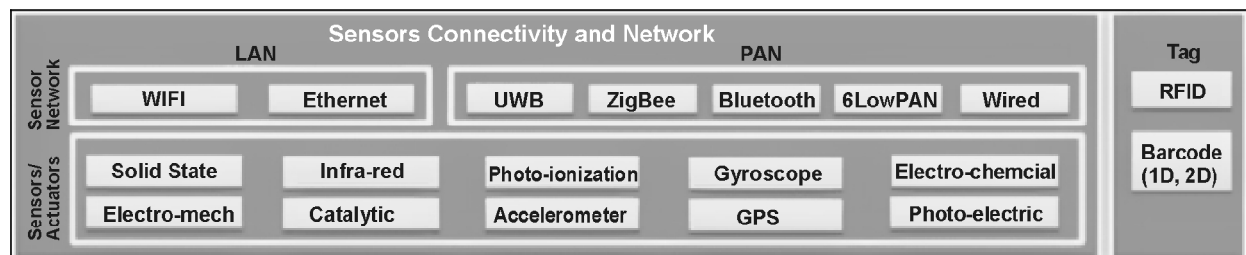


Fig. : Sensor, Connectivity and Network Layer

Gateway and Network Layer

- ▶ Gateways are responsible for routing the data coming from the **Sensor, Connectivity and Network layer** and pass it to the next layer which is the **Management Service Layer**.
- ▶ This layer requires having a large storage capacity for storing the enormous amount of data collected by the sensors, RFID tags etc. Also, this layer needs to have a consistently trusted performance in terms of public, private and hybrid networks.
- ▶ Different IoT device works on different kinds of network protocols. All this protocols are required to be assimilated in a single layer. This layer is responsible for integrating various network protocols.

From the figure below, at the bottom we have the gateway which is comprised of embedded OS, Signal Processors and Modulators, Micro-Controllers etc. Above the gateway we have the Gateway Networks which are LAN(Local Area Network), WAN(Wide Area Network) etc.

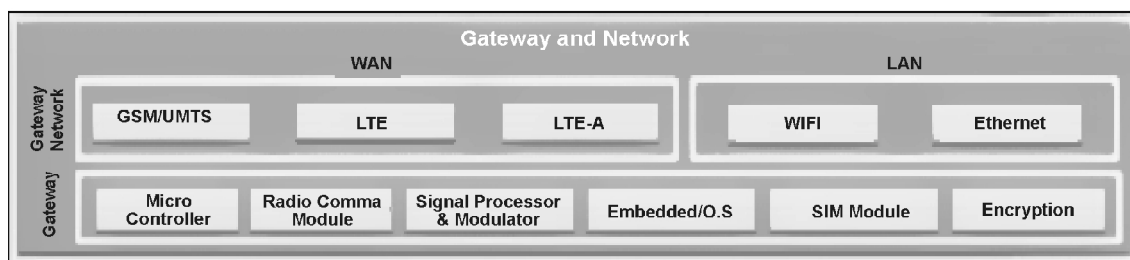


Fig. : Gateway and Network Layer

Management Service Layer

- ▶ This layer is used for managing the IoT services. Management Service layer is responsible for Securing Analysis of IoT devices, Analysis of Information (Stream Analytics, Data Analytics), Device Management.
- ▶ Data management is required to extract the necessary information from the enormous amount of raw data collected by the sensor devices to yield a valuable result of all the data collected. This action is performed in this layer.
- ▶ Also, certain situation requires immediate response to the situation. This layer helps in doing that by abstracting data, extracting information and managing the data flow.
- ▶ This layer is also responsible for data mining, text mining, service analytics etc.

From the figure below, we can see that, management service layer has Operational Support Service (OSS) which includes Device Modeling, Device Configuration and Management and many more. Also, we have the Billing Support System (BSS) which supports billing and reporting.

Also, from the figure, we can see that there are IoT/M2M Application Services which includes Analytics Platform; Data – which is the most important part; Security which includes Access Controls, Encryption, Identity Access Management etc. ; and then we have the Business Rule Management (BRM) and Business Process Management (BPM).

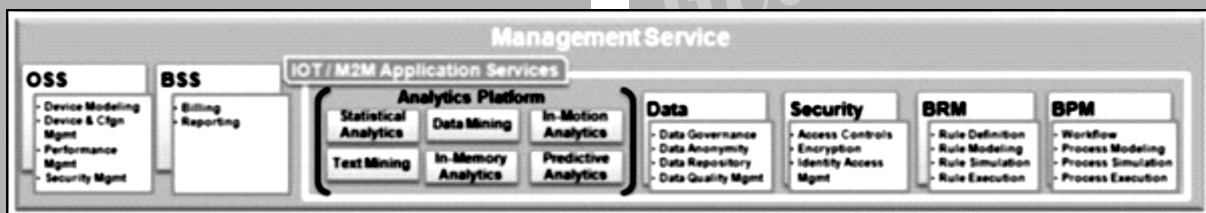


Fig. : Management Service Layer

Application Layer

Application layer forms the topmost layer of IoT architecture which are responsible for effective utilization of the data collected.

Various IoT applications include Home Automation, E-health, E-Government etc.

From the figure below, we can see that there are two types of applications which are Horizontal Market which includes Fleet Management, Supply Chain etc. and on the Sector wise application of IoT we have energy, healthcare, transportation etc.



Fig. : Application Layer

Smart Environment Application Domains

	Smart Home	Smart Office	Smart Retail	Smart City	Smart Agriculture	Smart Energy & Fuel	Smart Transportation	Smart Military
Network Size	Small	Small	Small	Medium	Medium /Large	Large	Large	Large
Network Connectivity	WPAN, WLAN, 3G, 4G, Internet	WPAN, WLAN, 3G, 4G, Internet	RFID, NFC, WPAN, WLAN, 3G, 4G, Internet	RFID, NFC, WLAN, 3G, 4G, Internet	WLAN, Satellite Comm., Internet	WLAN, 3G, 4G, Microwave links, Satellite Comm.	WLAN, 3G, 4G, Satellite Comm.	RFID, NFC, WPAN, WLAN, 3G, 4G, Satellite Comm.
Bandwidth Requirement	Small	Small	Small	Large	Medium	Medium	Medium~Large	Medium~Large

Fig. : Smart Environment Application Domains

where, **WLAN** stands for Wireless Local Area Network which includes Wi-Fi, WAVE, IEEE 802.11 a/b/g/p/n/ac/ad, and so on. **WPAN** stands for Wireless Personal Area Network which includes Bluetooth, ZigBee, 6LoWPAN, IEEE 802.15.4, UWB, and so on.

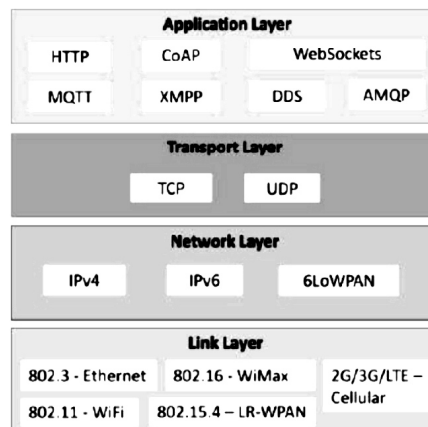
Service Domain	Services
Smart Home	Entertainment, Internet Access
Smart Office	Secure File Exchange, Internet Access, VPN, B2B
Smart Retail	Customer Privacy, Business Transactions, Business Security, B2B, Sales & Logistics Management
Smart City	City Management, Resource Management, Police Network, Fire Department Network, Transportation Management, Disaster Management
Smart Agriculture	Area Monitoring, Condition Sensing, Fire Alarm, Trespassing
Smart Energy & Fuel	Pipeline Monitoring, Tank Monitoring, Power Line Monitoring, Trespassing & Damage Management
Smart Transportation	Road Condition Monitoring, Traffic Status Monitoring, Traffic Light Control, Navigation Support, Smart Car Support, Traffic Information Support, ITS (Intelligent Transportation System)
Smart Military	Command & Control, Communications, Sensor Network, Situational Awareness, Security Information, Military Networking

Fig. : Smart Environment Application Domains: Service Domain and their Services classified.

Q3. Write about IoT protocol stack.

Ans.:

IoT Protocols



Link Layer

Link layer protocols basically determine how the data is sent over the network's physical layer or medium. The Hosts on the same link will exchange the data using these protocols. It also determines how the packets are coded and signaled by the hardware device over the medium to which the host is attached. Some of the link layer protocols are as follows: 802.3 Ethernet: 802.3 which represents a collection of wired Ethernet standards for the link layer.

Some examples :

- ▶ 802.3 – Ethernet (10Mbps – 40Gbps, coaxial, twisted pair)
- ▶ 802.11 – WiFi (1Mbps – 6,75 Gbps, wireless)
- ▶ 2G/3G/4G – Mobile Communications
- ▶ 802.16 – WiMax (1.5 Mbps – 1 Gbps, wireless)
- ▶ 802.15.1 – Bluetooth (1 Mbps, wireless)
- ▶ 802.15.4 – LR-WPAN (40 Kbps – 250 Kbps, wireless)

Network/Internet Layer

This layer is responsible for transmitting the information or the data from source to destination, by performing the host addressing and packet routing. The host identification can be done by hierarchical IP addressing schemes such as IPv4 or IPV6.

- 1) **IPV4** : Most deployed Internet protocol which is basically used for identifying devices in the network using hierarchical scheme. It uses 32 bit address scheme which allows 2³² or 4,294,967,296 addresses. No guaranteed delivery.
- 2) **IPV6** : Successor of IPV4 uses 128 bit scheme which allows 2¹²⁸ or 3.4 * 10³⁸ addresses.
- 3) **6LowPAN** : 6LowPAN (IPV6 Low power wireless Personal Area Networks) which brings IP protocols to the lower power devices that have limited processing capability, operates on 2.4GHz frequency range and provides data rates of 250 Kb/s.

These protocols also define compression mechanisms for IPV6 datagrams.

Transport Layer Protocols

These transport-layer protocols are responsible for the complete delivery of message transformation. The capability of message transfers can be set up on connections using TCP by handshaking or without acknowledgement as in UDP.

- 1) **TCP** : TCP is basically a Connection oriented protocol, ensures guaranteed delivery, provides error detection capability, avoids duplicate packets, provides flow control, congesting control to avoid congestion collapse improves network performance.
- 2) **UDP** : UDP is connectionless, basically used for the time-sensitive applications which have small data units for exchange, UDP is considered as transaction oriented and stateless protocol. Does not provide any guarantee on delivery and ordering of data transmission and even does not provide guarantee on duplication.

Application Layer Protocols

Application layer protocols basically define how the applications interface with the lower layer protocols to send the data over the network. Application layer protocols enable process-to-process connections by using the ports.

- 1) **HTTP** : Hypertext transfer protocol is a stateless protocol and, the protocol basically follows a request response model, each http request is independent of the other requests. HTTP uses universal resource identifiers to identify http resources.
- 2) **COAP** : Constrained Application protocol is designed to M2M(machine) applications. The CoAP is a online transmission protocol for usage with constrained nodes and constrained networks in IOT. The protocol is designed for M2M applications such as smart energy and building-automation. CoAP runs on connectionless-UDP rather than connection-oriented-TCP, uses Client server architecture for transmission.

- 3) **WEBSOCKET** : Web socket protocol uses full duplex communication over single socket connection to exchange the messages between client and server. It basically uses TCP, the stream of messages is sent between to and fro between client and server and the connection is kept open. Client can be an IOT device, browser or any mobile application.
- 4) **MQTT** : Message Queue Telemetry Transport relays on publish-subscribe model, it is a light weight messaging protocol. It uses client server architecture where client is an IOT device connects to server (MQTT broker) and then publishes the messages to topics on the server. The broker forwards the messages to clients subscribed to topics. The broker forwards the messages to the clients by subscribing to the topics. MQTT is suitable for limited processing devices and when the network bandwidth is low.
- 5) **XMPP** : Extensible Messaging and Presence Protocol (XMPP) is used for communication in realtime and streaming XML data between network entities. XMPP covers wide range of applications like data syndication, messaging, gaming, multiparty chat and voice/video calls. XMPP is considered as decentralized protocol which supports two way communication between client and server. XMPP allows communication between different IOT devices.
- 6) **DDS** : Data Distribution Service (DDS) is a middleware standard considered as data-centric which is basically used for device to device communication or machine to machine communication. DDS basically uses publish subscribers model, the topics were created by publishers for which subscribers can subscribe .The publisher is basically an object which is responsible for the distribution of data and subscriber is responsible for receiving published data. DDS provides quality of service control and also configurable reliability.

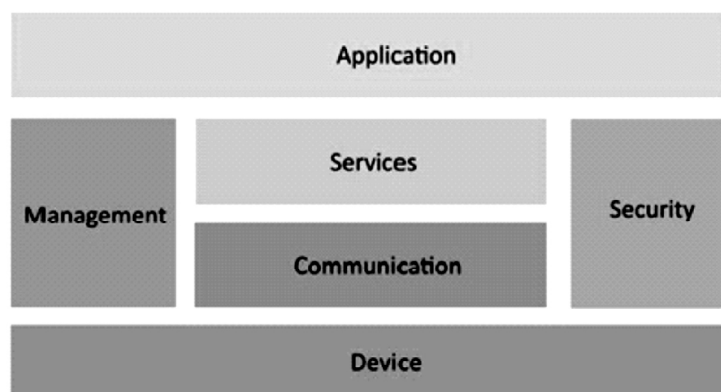
1.1.3 Logical Design of IOT

Q4. Explain various communication models of IoT.

Ans :

Logical design of an IoT system refers to an abstract representation of the entities and processes without going into the low-level specifics of the implementation.

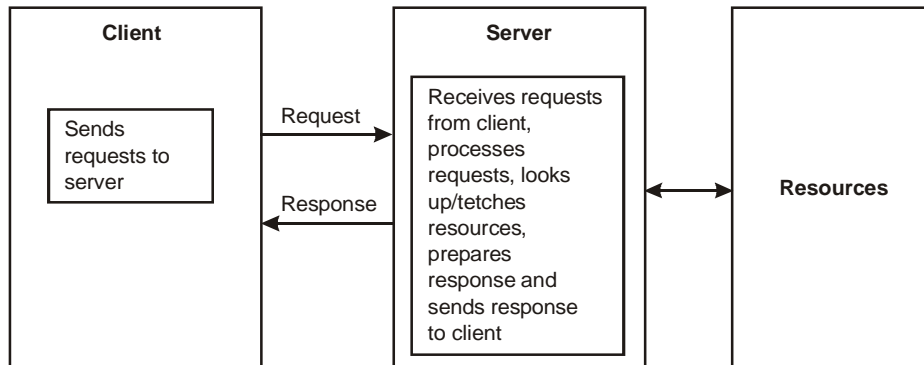
- An IoT system comprises of a number of functional blocks that provide the system the capabilities for identification, sensing, actuation, communication, and management.



Request-Response Communication Model

Request-Response is a communication model in which the client sends requests to the server and the server responds to the requests.

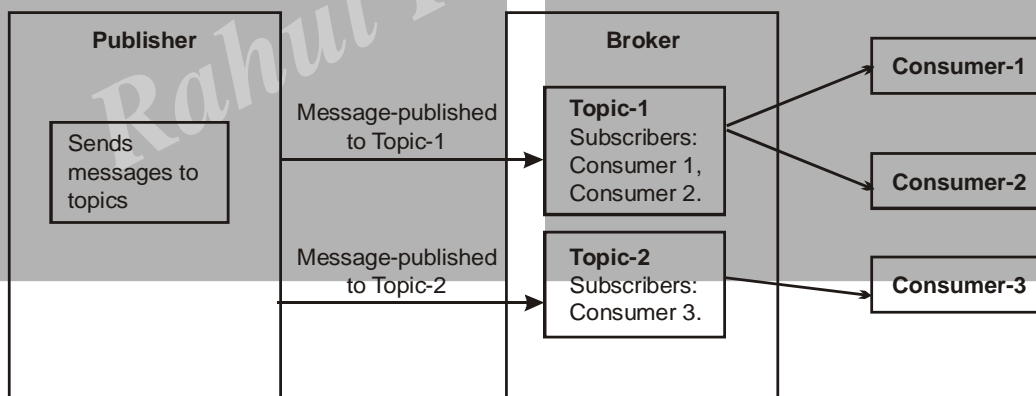
- ▶ When the server receives a request, it decides how to respond, fetches the data, retrieves resource representations, prepares the response, and then sends the response to the client.



Publish-Subscribe communication model

Publish-Subscribe is a communication model that involves publishers, brokers and consumers.

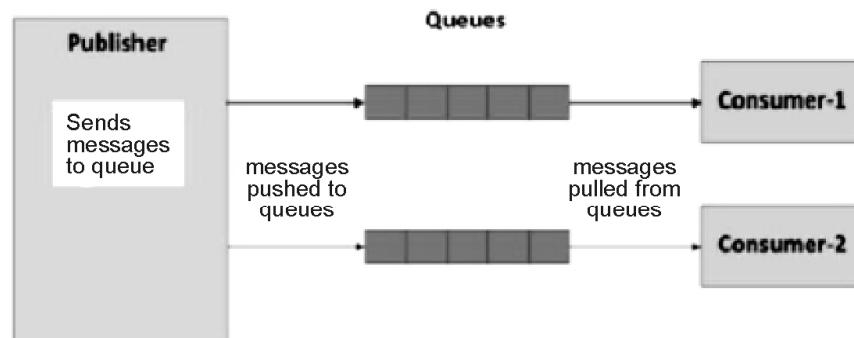
- ▶ Publishers are the source of data. Publishers send the data to the topics which are managed by the broker. Publishers are not aware of the consumers.
- ▶ Consumers subscribe to the topics which are managed by the broker.
- ▶ When the broker receives data for a topic from the publisher, it sends the data to all the subscribed consumers.



Push-Pull Communication Model

It is a communication model in which the data producers push the data to queues and the consumers pull the data from the queues. Producers do not need to be aware of the consumers.

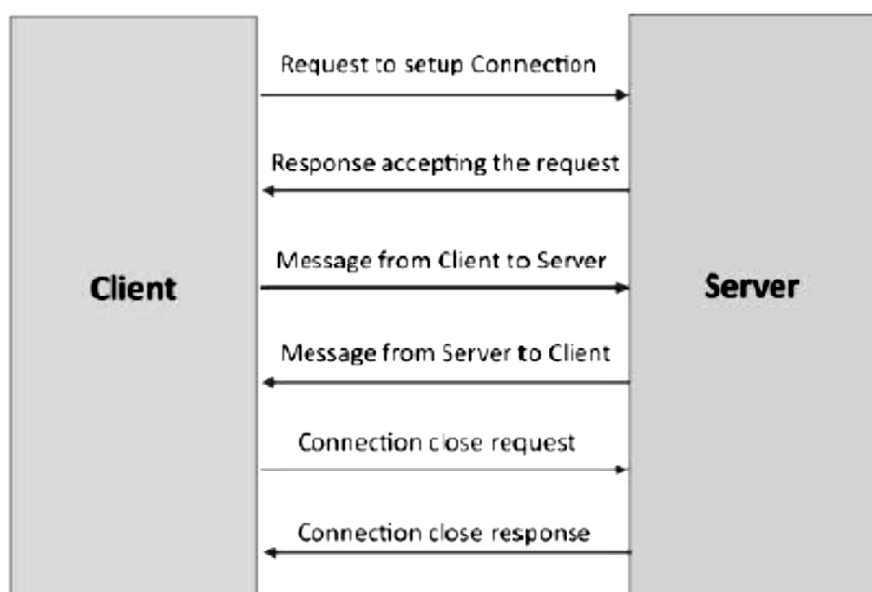
- ▶ Queues help in decoupling the messaging between the producers and consumers.
- ▶ Queues also act as a buffer which helps in situations when there is a mismatch between the rate at which the producers push data and the rate at which the consumers pull data.



Exclusive Pair communication model

Exclusive Pair is a bidirectional, fully duplex communication model that uses a persistent connection between the client and server.

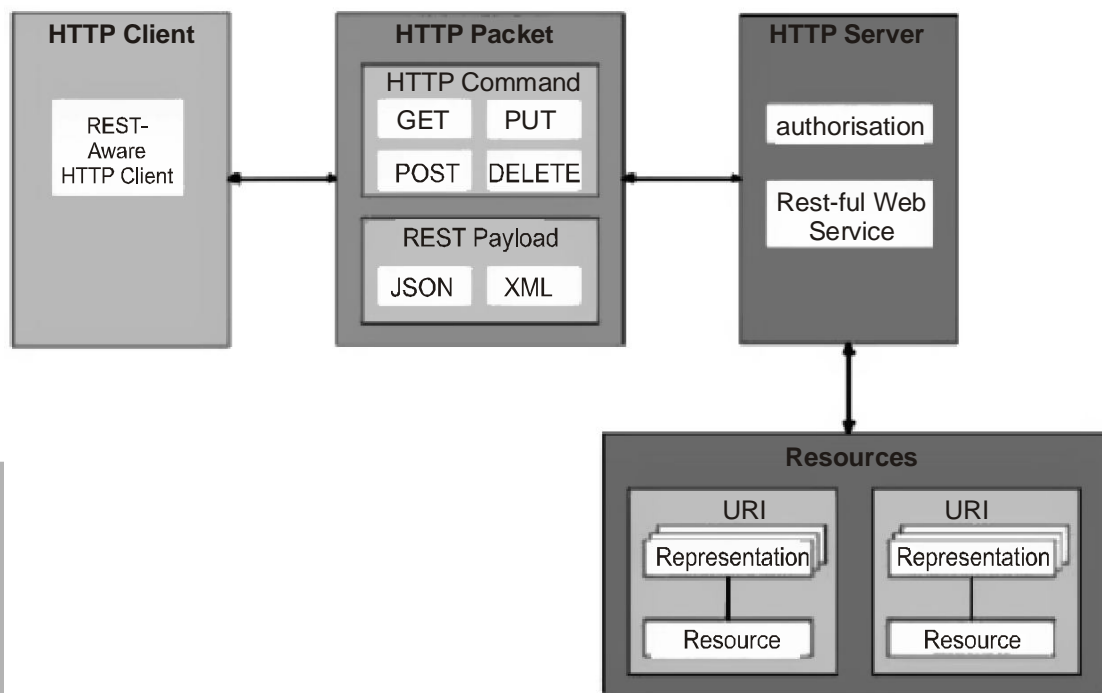
- Once the connection is setup it remains open until the client sends a request to close the connection.
- Client and server can send messages to each other after connection setup.



REST-based Communication APIs

Representational State Transfer (REST) is a set of architectural principles by which you can design web services and web APIs that focus on a system's resources and how resource states are addressed and transferred.

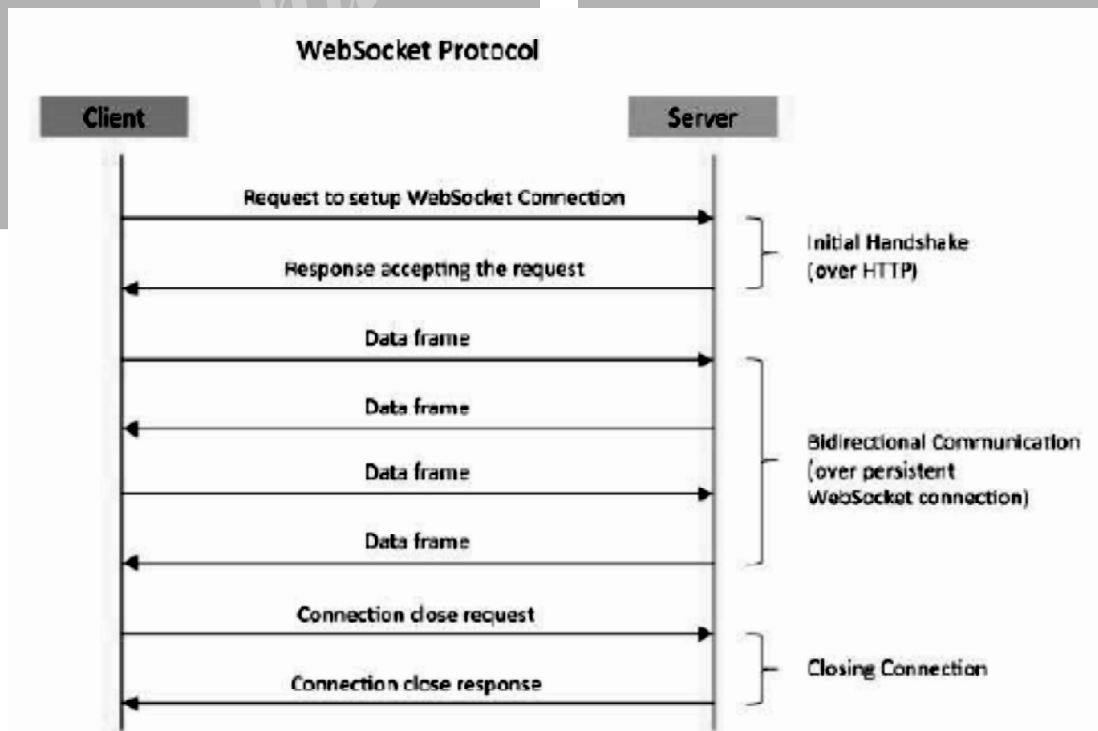
- REST APIs follow the request response communication model.
- The REST architectural constraints apply to the components, connectors, and data elements, within a distributed hypermedia system.



WebSocket-based Communication APIs

WebSocket APIs allow bidirectional, full duplex communication between clients and servers.

- ▶ WebSocket APIs follow the exclusive pair communication model.



1.1.4 IOT Enabling Technologies

Q5. Explain about various IoT Enabling technologies.

Ans :

Internet of things (IoT) is a global infrastructure for the information society, enabling advanced services by interconnecting (physical and virtual) things based on existing and evolving interoperable information and communication technologies. With the Internet of Things the communication is extended via Internet to all the things that surround us. The Internet of Things is much more than machine to machine communication, wireless sensor networks, sensor networks, 2G/3G/4G, GSM, GPRS, RFID, WI-FI, GPS, microcontroller, microprocessor etc.

A. BIG DATA

As more things (or "smart objects") are connected to the IoT, more data is collected from them in order to perform analytics to determine trends and associations that lead to insights. For example, an oil well equipped with 20-30 sensors can generate 500,000 data points every 15 second, a jetliner with 6,000 sensors generates 2.5 terabytes of data per day, and the more than 46 million smart utility meters installed in the U.S. generate more than 1 billion data points each day. Thus, the term "big data" refers to these large data sets that need to be collected, stored, queried, analyzed and generally managed in order to deliver on the promise of the IoT — insight!

Further compounding the technical challenges of big data is the fact that IoT systems must deal with not only the data collected from smart objects, but also ancillary data that is needed to properly perform such analytics (e.g., public and private data sets related to weather, GIS, financial, seismic, map, GPS, crime, etc.). Thus, as more smart objects come online, at least three metrics ("the three V's") are typically used by IoT operators to describe the big data they handle: volume (i.e., the amount of data they collect from their IoT sensors measured in gigabytes, terabytes and petabytes); velocity (i.e., the speed at which data is collected from the sensors); and variety (i.e., the different types of structured and unstructured data collected,

especially when compared to video and picture files as is typical within the consumer Internet).

B. DIGITAL TWIN

Another consequence of the growing and evolving IoT is the concept of a "digital twin," The concept refers to a digital copy of a physical asset (i.e., a smart object within the IoT), that lives and evolves in a virtual environment over the physical asset's lifetime. That is, as the sensors within the object collect real-time data, a set of models forming the digital twin is updated with all of the same information. Thus, an inspection of the digital twin would reveal the same information as a physical inspection of the smart object itself – albeit remotely. The digital twin of the smart object can then be studied to not only optimize operations of the smart object through reduced maintenance costs and downtime, but to improve the next generation of its design.

C. CLOUD COMPUTING

As the word "cloud" is often used as a metaphor for the Internet, "cloud computing" refers to being able to access computing resources via the Internet rather than traditional systems where computing hardware is physically located on the premises of the user and any software applications are installed on such local hardware.

Cloud computing – and its three service models of Software as a Service (SaaS), Platform as a Service (PaaS) and Infrastructure as a Service (IaaS) – are important to the IoT because it allows any user with a browser and an Internet connection to transform smart object data into actionable intelligence. That is, cloud computing provides "the virtual infrastructure for utility computing integrating applications, monitoring devices, storage devices, analytics tools, visualization platforms, and client delivery... [to] enable businesses and users to access [IoT-enabled] applications on demand anytime, anyplace and anywhere."

D. SENSORS

Central to the functionality and utility of the IoT are sensors embedded in smart objects. Such sensors are capable of detecting events or changes in a specific quantity (e.g., pressure), communicating the event or change data to the cloud (directly or via a gateway) and, in some

circumstances, receiving data back from the cloud (e.g., a control command) or communicating with other smart objects. Since 2012, sensors have generally shrunk in physical size and thus have caused the IoT market to mature rapidly. More specifically: "Technological improvements created microscopic scale sensors, leading to the use of technologies like Micro electromechanical systems (MEMS). This meant that sensors were now small enough to be embedded into unique places like clothing or other [smart objects]."

E. COMMUNICATIONS

With respect to sending and receiving data, wired and wireless communication technologies have also improved such that nearly every type of electronic equipment can provide data connectivity. This has allowed the ever-shrinking sensors embedded in smart objects to send and receive data over the cloud for collection, storage and eventual analysis.

The protocols for allowing IoT sensors to relay data include wireless technologies such as RFID, NFC, Wi-Fi, Bluetooth, Bluetooth Low Energy (BLE), XBee, ZigBee, Z-Wave, Wireless M-Bus, SIGFOX and NueINET, as well as satellite connections and mobile networks using GSM, GPRS, 3G, LTE, or WiMAX.

F. ANALYTICS SOFTWARE

Within the IoT ecosystem, Application Service Providers (ASPs) – which may or may not be from the companies who sell and service the smart objects – provide software to companies that can transform "raw" machine (big) data collected from smart objects into actionable intelligence (or insight). Generally speaking, such software performs data mining and employs mathematical models and statistical techniques to provide insight to users. That is, events, trends and patterns are extracted from big data sets in order to present the software's end-

users with insight in the form of portfolio analysis, predictions, risk analysis, automations and corrective, maintenance and optimization recommendations. In many cases, the ASPs may provide general analytical software or software targeting specific industries or types of smart objects.

G. EDGE DEVICES

The smart objects embedded with sensors connect via the Internet to the various service provider systems. The answer is via "edge devices" – any device such as a router, routing switch, integrated access device (IAD), multiplexer, or metropolitan area network (MAN) and wide area network (WAN) access device which provides an entry point from the global, public Internet into an ASP's or other enterprise's private network.

These edge devices are becoming smarter at processing data before such data even reaches an enterprise network's backbone (i.e., its core devices and cloud data centers).

1.1.5 IOT Levels & Deployment Templates

Q6. What are the various deployment templates of IoT ?

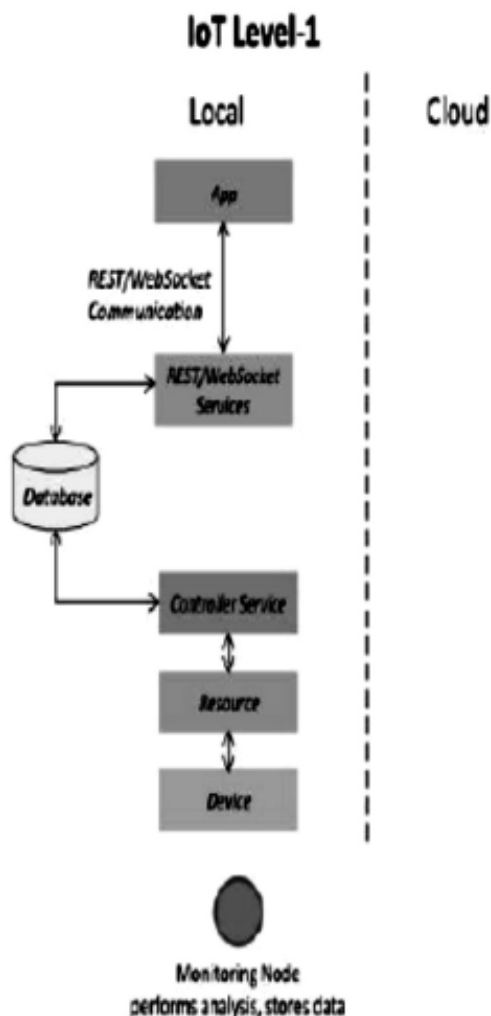
Ans :

Database: Database can be either local or in the cloud and stores the data generated by the IoT device.

- ▶ **Web Service :** Web services serve as a link between the IoT device, application, database and analysis components. Web service can be either implemented using HTTP and REST principles (REST service) or using WebSocket protocol (WebSocket service).
- ▶ **Analysis Component :** The Analysis Component is responsible for analysing the IoT data and generate results in a form which are easy for the user to understand.
- ▶ **Application :** IoT applications provide an interface that the users can use to control and monitor various aspects of the IoT system. Applications also allow users to view the system status and view the processed data.

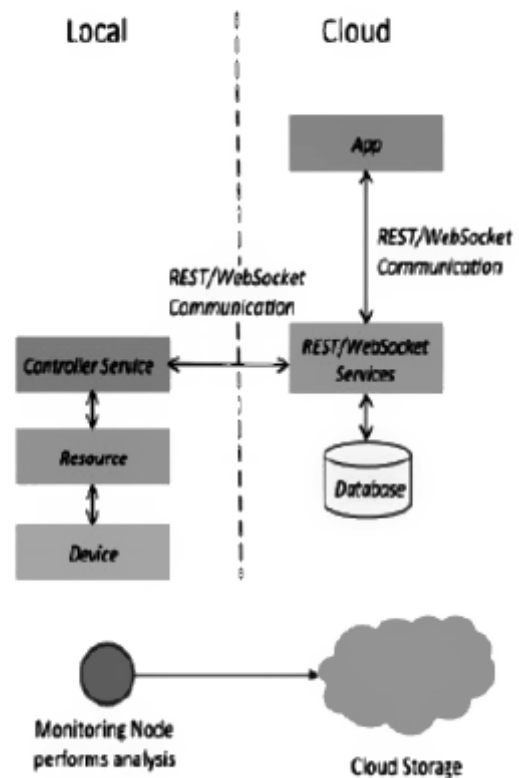
Q7. Explain the deployment levels of IoT.*Ans :***IoT Level-1**

- ▶ A level-1 IoT system has a single node/device that performs sensing and/or actuation, stores data, performs analysis and hosts the application
- ▶ Level-1 IoT systems are suitable for modelling low cost and low-complexity solutions where the data involved is not big and the analysis requirements are not computationally intensive.

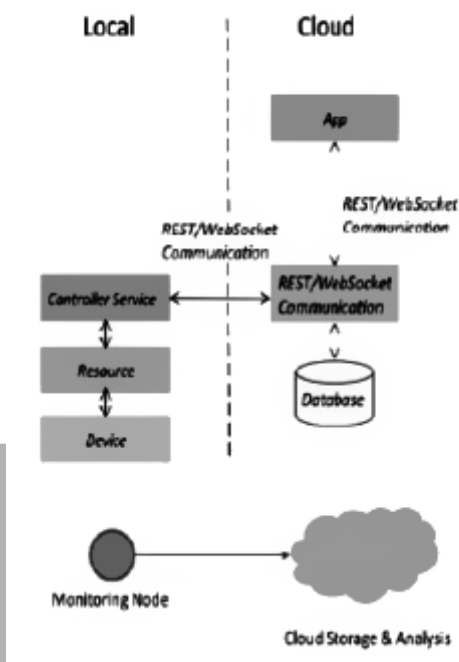
**IoT Level-2**

A level-2 IoT system has a single node that performs sensing and/or actuation and local analysis.

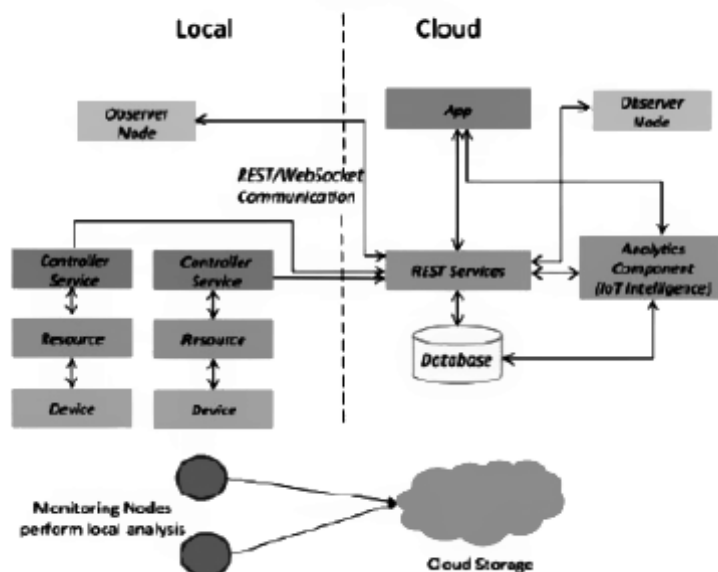
- ▶ Data is stored in the cloud and application is usually cloud based.
- ▶ Level-2 IoT systems are suitable for solutions where the data involved is big, however, the primary analysis requirement is not computationally intensive and can be done locally itself.

IoT Level-2**IoT Level-3**

- ▶ A level-3 IoT system has a single node. Data is stored and analysed in the cloud and application is cloud based.
- ▶ Level-3 IoT systems are suitable for solutions where the data involved is big and the analysis requirements are computationally intensive.

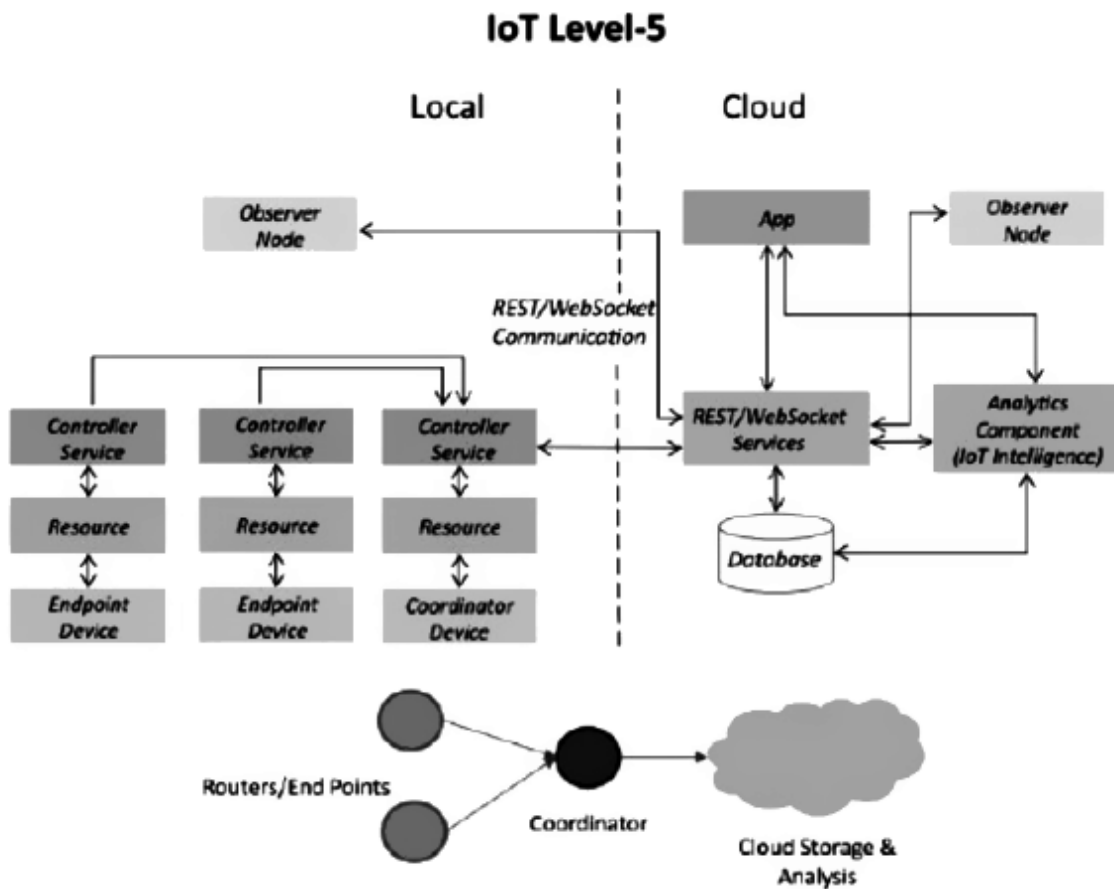
IoT Level-3**IoT Level-4**

- ▶ A level-4 IoT system has multiple nodes that perform local analysis. Data is stored in the cloud and application is cloud-based.
- ▶ Level-4 contains local and cloud based observer nodes which can subscribe to and receive information collected in the cloud from IoT devices.
- ▶ Level-4 IoT systems are suitable for solutions where multiple nodes are required, the data involved is big and the analysis requirements are computationally intensive.

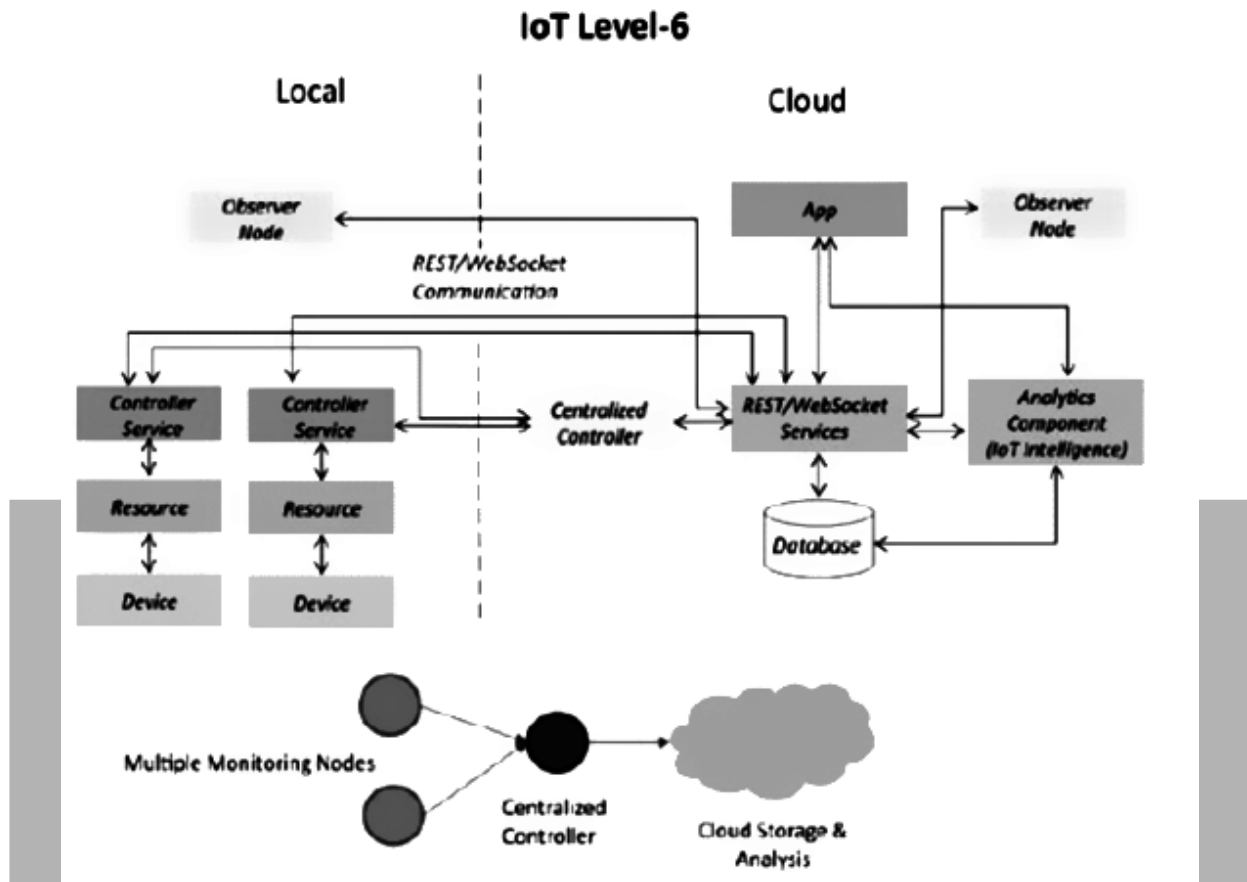
IoT Level-4

IoT Level-5

- ▶ A level-5 IoT system has multiple end nodes and one coordinator node.
- ▶ The end nodes that perform sensing and/or actuation.
- ▶ Coordinator node collects data from the end nodes and sends to the cloud.
- ▶ Data is stored and analyzed in the cloud and application is cloud-based.
- ▶ Level-5 IoT systems are suitable for solutions based on wireless sensor networks, in which the data involved is big and the analysis requirements are computationally intensive.

**IoT Level-6**

- ▶ A level-6 IoT system has multiple independent end nodes that perform sensing and/or actuation and send data to the cloud.
- ▶ Data is stored in the cloud and application is cloud-based.
- ▶ The analytics component analyses the data and stores the results in the cloud database.
- ▶ The results are visualized with the cloud-based application.
- ▶ The centralized controller is aware of the status of all the end nodes and sends control commands to the nodes.



1.2 DOMAIN SPECIFIC IOTS

1.2.1 Home Automation

Q8. Write about various IoT based Home Automation technologies.

Ans :

► Smart Lighting

Smart lighting for homes helps in saving energy by adapting the lighting to the ambient conditions and switching on/off or dimming the lights when needed. Key enabling technologies for smart lighting include solid state lighting (such as LED lights) and IP-enabled lights. For solid state lighting solutions both spectral and temporal characteristics can be configured to adapt illumination to various needs. Smart lighting solutions for home achieve energy savings by sensing the human movements and their environments and controlling the lights accordingly. Wireless-enabled and Internet connected lights can be controlled remotely from IoT applications such as a mobile or web application.

Smart lights with sensors for occupancy, temperature, lux level etc.. can be configured to adapt the lighting (by changing the light intensity, color, etc.) based on the ambient conditions sensed, in order to provide a good ambiance.

► Smart Appliances

Modern homes have a number of appliances such as TVs, refrigerator, music systems, washer/dryers, etc. Managing and controlling these appliances can be cumbersome, with each appliance having

its Wifi controls or remote controls. Smart appliances make the management easier and also provide status information to the users remotely. For example, smart washer/dryers that can be controlled remotely and notify when the washing/drying cycle is complete. Smart thermostats allow controlling the temperature remotely and can learn the user preferences.

Smart refrigerators can keep track of the items stored (using RFID tags) and send updates to the users when an item is low on stock .

Smart TVs allows users to search and stream videos and movies from the Internet on a local storage drive, search TV channel schedules and fetch news, weather updates and other content from the Internet.

► **Intrusion Detection**

Home intrusion detection systems use security cameras and sensors (such as PIR sensors and door sensors) to detect intrusions and raise alerts. Alerts can be in the form of SMS or an email sent to the user.

Advanced systems can even send detailed alerts such as an image grab or a short video clip sent as an email attachment.

A cloud controlled intrusion detection system is described in that uses location-aware services, where the geo-location of each node of a home automation system is independently detected and stored in the cloud.

Smoke detectors are installed in homes and buildings to detect smoke that is typically an early Sign of fire.

1.2.2 Cities

Q9. Explain about various IoT technologies used for smart cities.

Ans :

► **Smart Parking**

Finding a parking space during rush hours in crowded cities can be time consuming and frustrating. Furthermore, drivers blindly searching for parking spaces create additional traffic congestion.

Smart parking make the search for parking space easier and convenient for drivers. Smart parking are powered by IoT systems that detect the number of empty parking slots and send the information over the Internet to smart parking application hack-ends.

These applications can be accessed by the drivers from smart-phones. tablets and in-car navigation systems.

In smart parking, sensors are used (for each parking slot, to detect whether the slot is empty or occupied. This information is aggregated by a local controller and then sent over the Internet to the database.

► **Smart Lighting**

Smart lighting systems for roads, parks and buildings can help in saving energy. According to an IEA report, lighting is responsible for 19% of global electricity use and around 6% of global greenhouse gas emissions.

Smart lighting allows lighting to be dynamically controlled and also adaptive to the ambient conditions. Smart lights connected to the Internet can be controlled remotely to configure lighting schedules and lighting intensity. Custom lighting configurations can be set for different situations such as a foggy day, a festival. etc.

Smart lights equipped with sensors can communicate with other lights and exchange information on the sensed ambient conditions to adapt the lighting.

► **Smart Roads**

Smart roads equipped with sensors can provide information on driving conditions, travel time estimates and alerts in case of poor driving conditions. traffic congestions and accidents. Such information can help in making the roads safer and help in reducing traffic jams.

Information sensed from the roads can be communicated via Internet to cloud—based applications and social media and disseminated to the drivers who subscribe to such applications.

The system can provide the drivers and passengers with a consistent view of the road situation a few hundred meters ahead of them.

► **Structural Health Monitoring**

Structural health Monitoring systems use a network of sensors to monitor the vibration levels in the structures such as bridges and buildings. The data collected from these sensors is analyzed to assess the health of the structures.

By analyzing the data it is possible to detect cracks and mechanical breakdowns, locate the damages to a structure and also calculate the remaining life of the structure.

Using such systems, advance warnings can be given in the case of imminent failure of the structure.

► **Surveillance**

Surveillance of infrastructure, public transport and events in cities is required to ensure safety and security.

City wide surveillance infrastructure comprising of large number of distributed and Internet connected video surveillance cameras can be created.

The video feeds from surveillance cameras can be aggregated in cloud-based scalable storage solutions. Cloud-based video analytics applications can be developed to search for patterns or specific events from the video feeds.

► **Emergency Response**

IoT systems can be used for monitoring the critical infrastructure in cities such as buildings, gas and water pipelines, public transport and power substations. IoT systems for fire detection, gas and water leakage detection can help in generating alerts and minimizing their effects on the critical infrastructure, IoT systems for critical infrastructure monitoring enable aggregation and sharing of information collected from large number of sensors.

Using cloud-based architectures, multi-modal information such as sensor data, audio, video feeds can be analyzed in near real-time to detect adverse events.

Response to alerts generated by such systems can be in the form of alerts sent to the public, re-routing of traffic, evacuations of the affected areas, etc.

1.2.3 Environment

Q10. Write Which IoT technologies are used for Environmental Monitoring.

Ans :

► **Weather Monitoring**

IoT-based weather monitoring systems can collect data from a number of sensor attached (such as temperature, humidity, pressure, etc.) and send the data to cloud-based applications and storage back-ends.

The data collected in the cloud can then be analyzed and visualized by cloud-based applications. Weather alerts can be sent to the subscribed users from such applications.

AirPi is a weather and air quality monitoring kit capable of recording and uploading information about temperature, humidity, air pressure, light levels, UV levels, carbon monoxide, nitrogen dioxide and smoke level to the Internet.

► **Air Pollution Monitoring**

IoT based air pollution monitoring systems can monitor emission of harmful gases (CO₂, CO, NO, NO_x, etc.) by factories and automobiles using gaseous and meteorological sensors. The collected data can be analyzed to make informed decisions on pollution control approaches.

► **Noise Pollution Monitoring**

Due to growing urban development, noise levels in cities have increased and even become alarmingly high in some cities. Noise pollution can cause health hazards for humans due to sleep disruption and stress. Noise pollution monitoring can help in generating noise maps for cities. Urban noise maps can help the policy makers in urban planning and making policies to control noise levels near residential areas, schools and parks.

IoT based noise pollution monitoring systems use a number of noise monitoring stations that are developed at different places in a city. The data on noise levels from the stations is collected on servers or in the cloud. The collected data is then aggregated to generate noise maps.

► Forest Fire Detection

Forest fires can cause damage to natural resources, property and human life. There can be different causes of forest fires including lightening, human negligence, volcanic eruptions and sparks from rock falls. Early detection of forest fires can help in minimizing the damage.

IoT based forest fire detection systems use a number of monitoring nodes deployed at different locations in a forest. Each monitoring node collects measurements on ambient conditions including temperature, humidity, light levels, etc.

► River Floods Detection

River floods can cause extensive damage to the natural and human resources and human life. River floods occur due to continuous rainfall which cause the river levels to rise and flow rates to increase rapidly. Early warnings of floods can be given by monitoring the water level and flow rate.

IoT based river flood monitoring system use a number of sensor nodes that monitor the water level (using ultrasonic sensors) and flow rate (using the flow velocity sensors).

Data from a number of such sensor nodes is aggregated in a server or in the cloud. Monitoring applications raise alerts when rapid increase in water level and flow rate is detected.

1.2.4 Energy

Q11. Write about Smart Energy devices.

Ans :

► Smart Grids

Smart Grid is a data communications integrated with the electrical grid that collects and analyzes data captured in near-real-time about power transmission, distribution, and consumption.

Smart Grid technology provides predictive information and recommendations to utilities, their suppliers, and their customers on how best to manage power. Smart grids collect data regarding electricity generation (centralized or distributed), consumption (instantaneous or predictive), storage (or conversion of energy into other forms), distribution and equipment health data.

Smart grids use high-speed, fully integrated, two-way communication technologies for real-time information and power exchange.

By using IoT based sensing and measurement technologies, the health of equipment and the integrity of the grid can be evaluated. Smart meters can capture almost real-time consumption, remotely control the consumption of electricity and remotely switch off supply when required.

Power thefts can be prevented using smart metering. By analyzing the data on power generation, transmission and consumption smart grids can improve efficiency throughout the electric system.

Storage collection and analysis of smart grids data in the cloud can help in dynamic optimization of system operations, maintenance, and planning. Cloud-based monitoring of smart grids data can improve energy usage levels via energy feedback to users coupled with real-time pricing information.

► Renewable Energy Systems

Due to the variability in the output from renewable energy sources (such as solar and wind), integrating them into the grid can cause grid stability and reliability problems. Variable output produces local voltage swings that can impact power quality.

Existing grids were designed to handle power flows from centralized generation sources to the loads through transmission and distribution lines. When distributed renewable energy sources are integrated into the grid, they create power bi-directional power flows for which the grids were not originally designed.

IoT based systems integrated with the transformers at the point of interconnection measure the electrical variables and how much power is fed into the grid. To ensure the grid stability, one solution is to simply cut off the overproduction. For wind energy systems, closed-loop controls can be used to regulate the voltage at point of interconnection which coordinate wind turbine outputs and provides reactive power support.

► Prognostics

Energy systems (smart grids, power plants, wind turbine farms, for instance) have a large

number of critical components that must function correctly so that the systems can perform their operations correctly.

For example, a wind turbine has a number of critical components, e.g., bearings, turning gears. for instance, that must be monitored carefully as wear and tear in such critical components or sudden change in operating conditions of the machines can result in failures. In systems such as power grids, real-time information is collected using specialized electrical sensors called Power Measurement Units at the substations.

The information received from PMUs intend to be monitored in real-time for estimating the state of the system and for predicting failures.

Energy systems have thousands of sensors that gather real-time maintenance data continuously for condition monitoring and failure.

1.2.5 Retail

Q12. Write about the technologies IoT used for smart retail industries.

Ans :

► **Inventory Management**

Inventory Management for retail has become increasingly important in the recent years with the growing competition. while over-stocking of products can result in additional storage expenses and risk. under-stocking can lead to loss of revenue.

IoT systems using Radio Frequency identification (RFID) tags can help in Inventory Management and maintaining the right inventory. RFID tags attached to the products allow them to be tracked in real time ,so that the inventory levels can be determined accurately and products which are low on stock can be replenished.

Tracking can be done using RFID readers attached to the retail store achieves or in the warehouse.

IoT systems enable remote monitoring of inventory using the data collected by the RFID readers.

► **Smart Payments**

Smart payment solutions such as contact-less payments powered by technologies such as Near field communication (NFC) and Bluetooth.

Near field communication (NFC) is a set of standards for smart-phones and other devices to communicate with each other by bringing them into proximity or by touching them. (customers can store the credit card information in their NFC-enabled smart-phones and make payments by bringing the smart-phones near the point of sale terminals. NFC may be used in combination with Bluetooth. where NFC initiates initial pairing of devices to establish a Bluetooth connection while the actual data transfer takes place over Bluetooth.

► **Smart Vending Machines**

Smart vending machines connected to the Internet allow remote monitoring of inventory levels, elastic pricing of products, promotions, and contact-less payments using NFC, Smart-phone applications that communicate with smart vending machines allow user preferences to be remembered and learned with time. When a user moves from The vending machine to the other and pairs the smart phone with the vending machine, a user specific interface is presented. Users can save their preferences and favorite products.

Sensors in a smart vending machine monitor its operations and send the data to the cloud which can be used for predictive maintenance, Smart vending machines can communicate with other vending machines in their vicinity levels so that time customers can be routed the smart vending machines can reduce the price as the expiry date nears.

1.2.6 Agriculture

Q13. Explain the Smart Agriculture technologies.

Ans :

► **Smart Irrigation**

Smart irrigation systems can improve crop yields while saving water. Smart irrigation systems use IoT devices with soil moisture sensors to determine the amount of moisture in the soil and

release the flow of water through the irrigation pipes only when the moisture levels go below a predefined threshold.

Smart irrigation systems also collect moisture level measurements on a server or in the cloud where the collected data can be analyzed to plan watering schedules.

Cultivar's RainCloud is a device for smart irrigation that uses water valves, soil sensors and a WiFi enabled programmable computer.

► **Green House Control**

Green houses are structures with glass or plastic roofs that provide conducive environment for growth of plants. The climatological conditions inside a green house can be monitored and controlled to provide the best conditions, for growth of plants.

The temperature, humidity, soil moisture, light and carbon dioxide levels are monitored using sensors and the climatological conditions are controlled automatically using actuation devices (such as valves for releasing water and switches for controlling fans), IoT systems play an important role in green house control and help in improving productivity.

The data collected from various sensors is stored on centralised servers or in the cloud where analysis is performed to optimise the control strategies and also correlate the productivity with different control strategies.

1.2.7 Health & Lifestyle

Q14. Write about which technologies are used by IoT for health and life style monitoring.

Ans :

► **Health & Fitness Monitoring**

Wearable IoT devices that allow non-invasive and continuous monitoring of physiological parameters can help in continuous health and fitness monitoring.

These wearable devices may can be in variation forms such as belts and wrist hands. The wearable devices form a type of wireless sensor networks called body area networks in which the measurements from a number of wearable devices are continuous sent to a master node (such as a smart-phone) which then sends the data to a server or a cloud-based back-end for analysis and archiving.

Health-care providers can analyze the collected health care data to determine any health conditions or anomalies.

Commonly uses body sensors include: body temperature, heart rate, pulse oximeter, oxygen saturation, blood pressure, electrocardiogram movement and electro-echophalogram (EEG, (ECG). accelerometer and oxygen saturation (SpO2) sensors.

► **Wearable Electronics**

Wearable electronics such as wearable gadgets (smart watches, smart glasses, Wristbands etc.) and fashion electronics (with electronics integrated in clothing and accessories. (e.g. (Google Glass or Moto 360 smart watches provide various functions and features to assist us in our daily activities and making us lead healthy lifestyles.

Smart watches that run mobile operating systems (such as Android) provide enhanced functionality beyond just timekeeping. With smart watches, the users can search the Internet, play audio/video files, play games etc.

UNIT II

IoT and M2M : Introduction to M2M, Difference between IoT and M2M, SDN and NFV for IoT. IoT System Management with NETCONF-YANG: Need for IoT Systems Management, SNMP, Network Operator requirements, NETCONF, YANG, IoT Systems Management with NETCONF-YANG. IoT Platforms Design Methodology: Introduction, IoT Design Methodology, Case Study on IoT system for weather Monitoring. Motivation for Using Python. Python Packages for IoT.

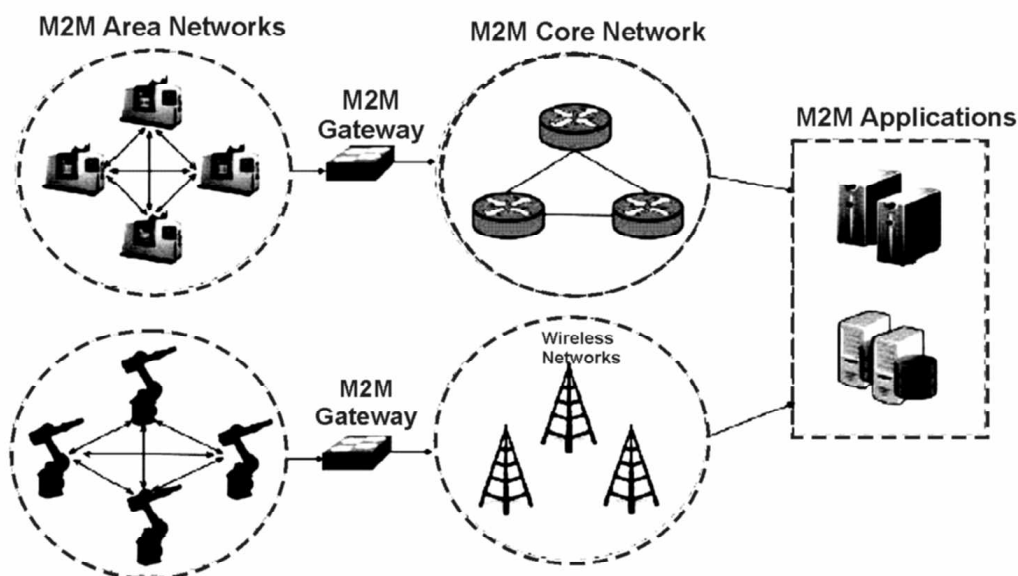
2.1 IOT AND M2M

2.1.1 Introduction To M2M

Q1. What is M2M communication? Write a note on them.

Ans :

Machine-to-Machine (M2M) communication is a form of data communication that involves one or more entities that do not necessarily require human interaction or intervention in the process of communication.



M2M is also named as Machine Type Communication (MTC) in 3GPP. It is different from the current communication models in the ways that it involves :

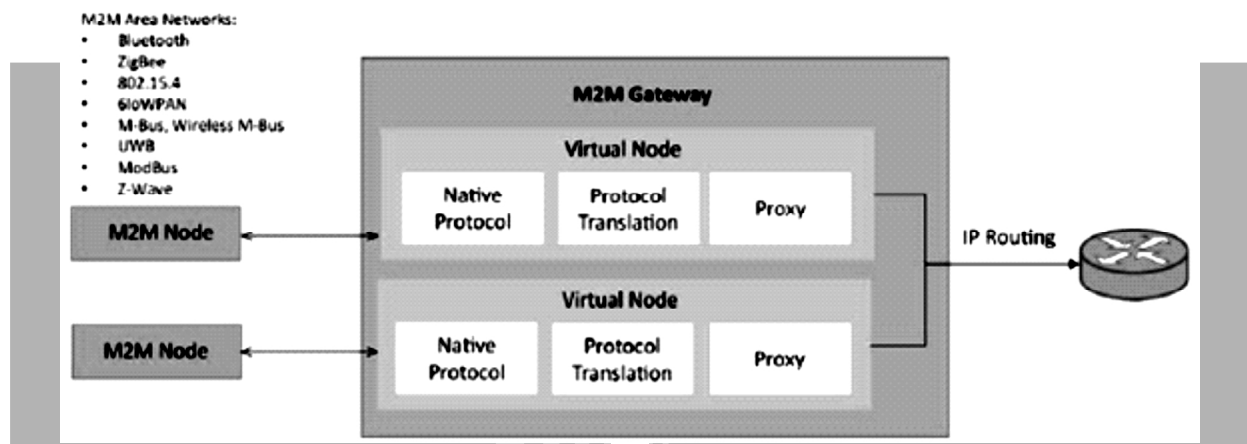
- ▶ new or different market scenarios
- ▶ lower costs and effort
- ▶ a potentially very large number of communicating terminals
- ▶ little traffic per terminal, in general.

M2M communication could be carried over mobile networks (e.g. GSM-GPRS, CDMA EVDO networks). In the M2M communication, the role of mobile network is largely confined to serve as a transport network.

With a potential market of probably 50 million connected devices, M2M offers tremendous opportunities as well as unique challenges. These devices vary from highly-mobile vehicles communicating in real-time, to immobile meter-reading appliances that send small amounts of data sporadically.

M2M Gateway

- ▶ Since non-IP based protocols are used within M2M area networks, the M2M nodes within one network cannot communicate with nodes in an external network.
- ▶ To enable the communication between remote M2M area networks, M2M gateways are used.



Q2. What are the applications and features of M2M communication ?

Ans :

Applications of M2M

The applications of M2M cover many areas and the areas in which M2M is currently used are given below :

- a. **Security** : Surveillances, Alarm systems, Access control, Car/driver security
- b. **Tracking & Tracing** : Fleet Management, Order Management, Pay as you drive, Asset Tracking, Navigation, Traffic information, Road tolling, Traffic optimization/steering
- c. **Payment** : Point of sales, Vending machines, Gaming machines
- d. **Health** : Monitoring vital signs, Supporting the aged or handicapped, Web Access Telemedicine points, Remote diagnostics
- e. **Remote Maintenance/Control** : Sensors, Lighting, Pumps, Valves, Elevator control, Vending machine control, Vehicle diagnostics
- f. **Metering** : Power, Gas, Water, Heating, Grid control, Industrial metering
- g. **Manufacturing** : Production chain monitoring and automation
- h. **Facility Management** : Home / building / campus automation

Key Features of M2M

Some of the key features of M2M communication system are given below :

- a. **Low Mobility** : M2M Devices do not move, move infrequently, or move only within a certain region
 - b. **Time Controlled** : Send or receive data only at certain pre-defined periods
 - c. **Time Tolerant** : Data transfer can be delayed
 - d. **Packet Switched** : Network operator to provide packet switched service with or without an MSISDN
 - e. **Online small Data Transmissions**: MTC Devices frequently send or receive small amounts of data.
 - f. **Monitoring**: Not intend to prevent theft or vandalism but provide functionality to detect the events.
 - g. **Low Power Consumption** : To improve the ability of the system to efficiently service M2M applications.
 - h. **Location Specific Trigger** : Intending to trigger M2M device in a particular area e.g. wake up the device.
-

2.1.2 Difference Between IoT And M2M

Q3. Differentiate between IoT and M2M communication.

Ans :

Communication Protocols

- ▶ M2M and IoT can differ in how the communication between the machines or devices happens.
- ▶ M2M uses either proprietary or non-IP based communication protocols for communication within the M2M area networks.

Machines in M2M vs Things in IoT

- ▶ The "Things" in IoT refers to physical objects that have unique identifiers and can sense and communicate with their external environment (and user applications) or their internal physical states.
- ▶ M2M systems, in contrast to IoT, typically have homogeneous machine types within an M2M area network.

Hardware vs Software Emphasis

- ▶ While the emphasis of M2M is more on hardware with embedded modules, the emphasis of IoT is more on software.

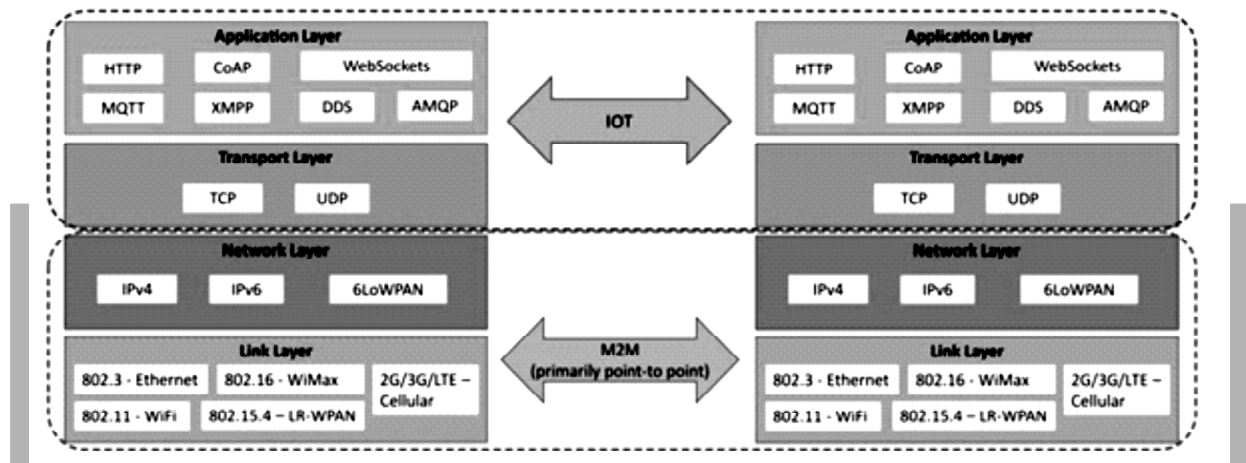
Data Collection & Analysis

- ▶ M2M data is collected in point solutions and often in on-premises storage infrastructure.
- ▶ In contrast to M2M, the data in IoT is collected in the cloud (can be public, private or hybrid cloud).

Applications

- ▶ M2M data is collected in point solutions and can be accessed by on-premises applications such as diagnosis applications, service management applications, and on premises enterprise applications.
- ▶ IoT data is collected in the cloud and can be accessed by cloud applications such as analytics applications, enterprise applications, remote diagnosis and management applications, etc.

Communication in IoT vs M2M



2.1.3 SDN and NFV for IOT

Q4. What is SDN ? Write about the role of SDN in IOT.

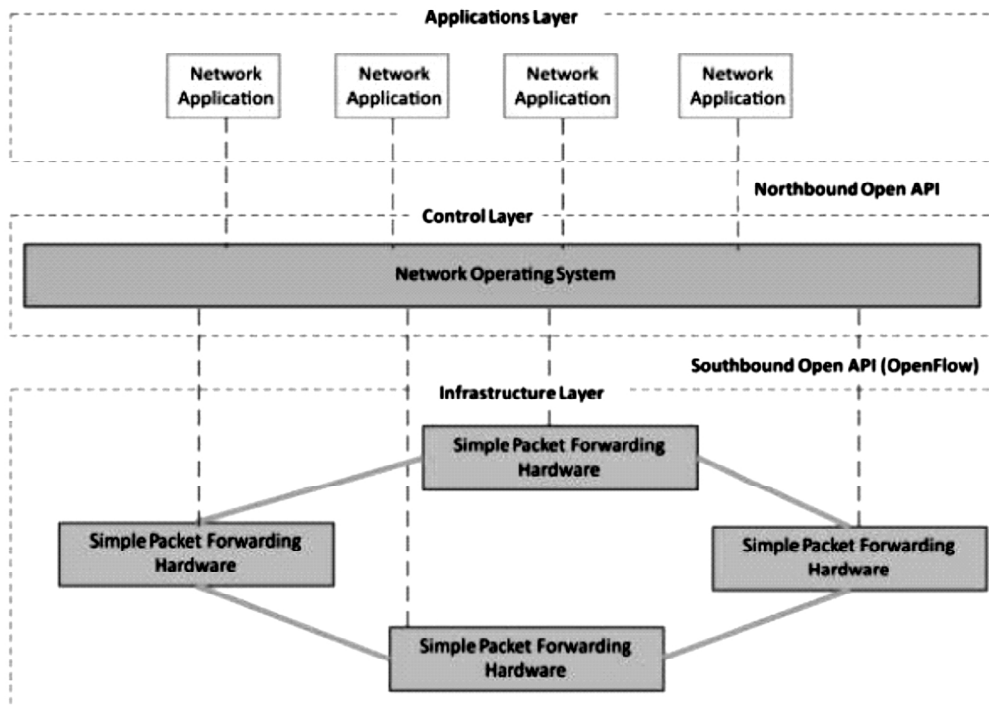
Ans :

SDN

Software-Defined Networking (SDN) is a networking architecture that separates the control plane from the data plane and centralizes the network controller.

The SDN Architecture is:

1. **Directly Programmable** : Network control is directly programmable because it is decoupled from forwarding functions.
2. **Agile** : Abstracting control from forwarding lets administrators dynamically adjust network-wide traffic flow to meet changing needs.
3. **Centrally Managed** : Network intelligence is (logically) centralized in software-based SDN controllers that maintain a global view of the network, which appears to applications and policy engines as a single, logical switch.
4. **Programmatically Configured** : SDN lets network managers configure, manage, secure, and optimize network resources very quickly via dynamic, automated SDN programs, which they can write themselves because the programs do not depend on proprietary software.
5. **Open Standards-based And Vendor-neutral** : When implemented through open standards, SDN simplifies network design and operation because instructions are provided by SDN controllers instead of multiple, vendor-specific devices and protocols.



Key elements of SDN

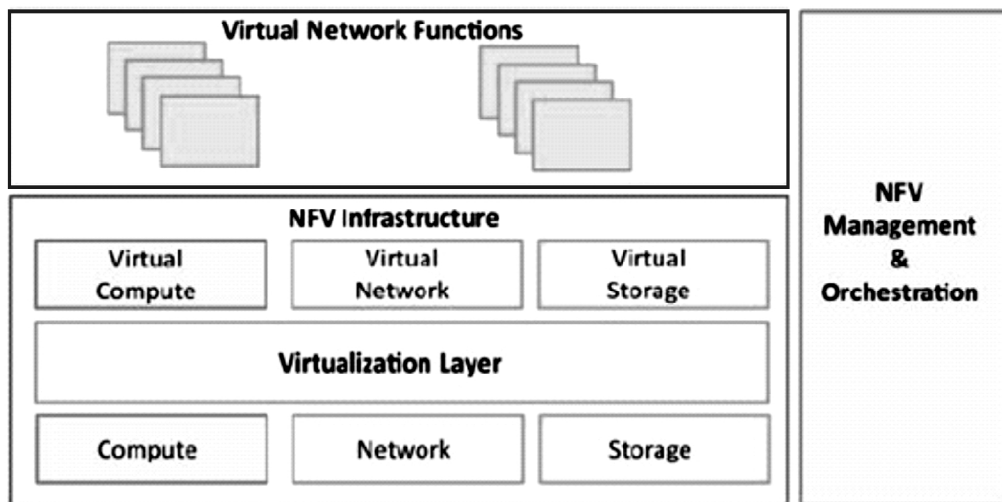
- ▶ Centralized Network Controller
- ▶ With decoupled control and data planes and centralized network controller, the network administrators can rapidly configure the network.
- ▶ Programmable Open APIs
- ▶ SDN architecture supports programmable open APIs for interface between the SDN application and control layers (Northbound interface).
- ▶ Standard Communication Interface (OpenFlow)
- ▶ SDN architecture uses a standard communication interface between the control and infrastructure layers (Southbound interface).
- ▶ OpenFlow, which is defined by the Open Networking Foundation (ONF) is the broadly accepted SDN protocol for the Southbound interface.

Q5. What is Network Function Virtualization ? Explain.

Ans :

NFV (Network Function Virtualization)

- ▶ Network Function Virtualization (NFV) is a technology that leverages virtualization to consolidate the heterogeneous network devices onto industry standard high volume servers, switches and storage.
- ▶ NFV is complementary to SDN as NFV can provide the infrastructure on which SDN can run.
- ▶ Because NFV architecture virtualizes network functions and eliminates specific hardware, network managers can add, move or change network functions at the server level in a simplified provisioning process.
- ▶ If a VNF running on a virtual machine requires more bandwidth, for example, the administrator can move the VM to another physical server or provision another virtual machine on the original server to handle part of the load. Having this flexibility allows an IT department to respond in a more agile manner to changing business goals and network service demands.

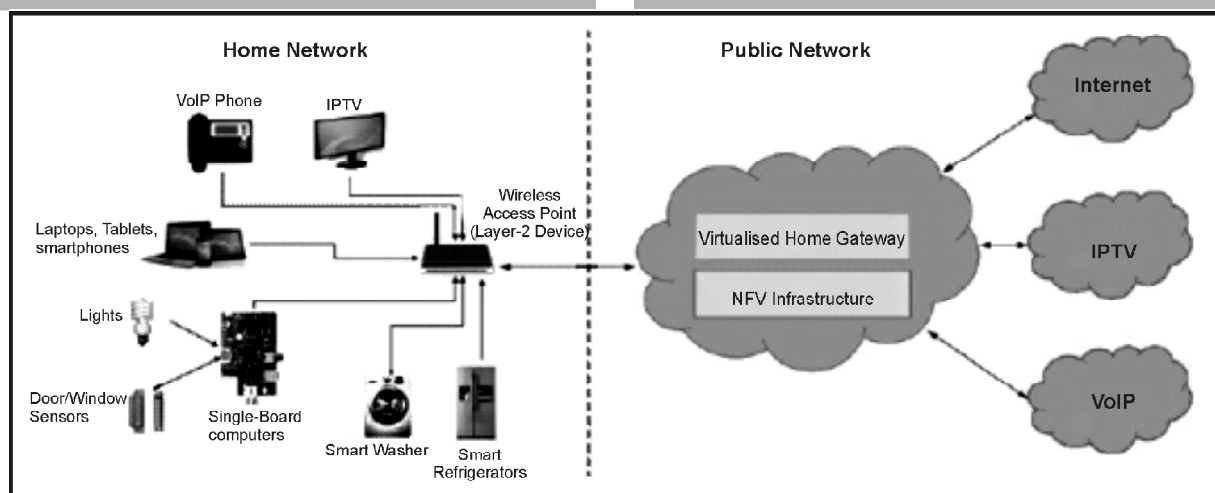


Key elements of NFV

- ▶ Virtualized Network Function (VNF):
- ▶ VNF is a software implementation of a network function which is capable of running over the NFV Infrastructure (NFVI).
- ▶ NFV Infrastructure (NFVI):
- ▶ NFVI includes compute, network and storage resources that are virtualized.
- ▶ NFV Management and Orchestration:
- ▶ NFV Management and Orchestration focuses on all virtualization-specific management tasks and covers the orchestration and life-cycle management of physical and/or software resources that support the infrastructure virtualization, and the life-cycle management of VNFs.

NFV Use Case

- ▶ NFV can be used to virtualize the Home Gateway. The NFV infrastructure in the cloud hosts a virtualized Home Gateway. The virtualized gateway provides private IP addresses to the devices in the home. The virtualized gateway also connects to network services such as VoIP and IPTV.



2.2 IOT SYSTEM MANAGEMENT WITH NETCONF-YANG

2.2.1 Need for IOT Systems Management

Q6. What is the need for IOT system Management? Explain.

Ans :

Need for IoT Systems Management

Internet of Things (IoT) systems can have complex software, hardware and deployment designs including sensors, actuators, software anti network resources, data collection and analysis services and user interfaces.

The need for managing IoT systems is described as follows :

- ▶ **Automating Configuration:** IoT system management capabilities can help in automating the system configurations. System management interfaces provide predictable and easy to use management capability and the ability to automate system configuration.
- ▶ **Monitoring Operational & Statistical Data:** Operational data is the data which is related to the system's operating parameters and is collected by the system at runtime. Statistical data is the data which describes the system performance (e.g. CPU and memory usage). Management systems can help in monitoring operational and statistical data of a system. This data can be used for fault diagnosis or prognosis.
- ▶ **Improved Reliability:** A management system that allows validating the system configurations before they are put into effect can help in improving the system reliability.
- ▶ **System Wide Configurations:** For IoT systems that consist of multiple devices or nodes, ensuring system wide configuration can be critical for the correct functioning of the system. Management approaches in which each device is configured separately (either through a manual or automated process) can result in system faults or undesirable outcomes. This happens when some devices are running on an old configuration while others start running on new configuration. To avoid this, system wide configuration is required where all devices are configured in a single atomic transaction. This ensures that the configuration changes are either applied to all devices or to none.
- ▶ **Multiple System Configurations:** For some systems it may be desirable to have multiple valid configurations which are applied at different times or in certain conditions.
- ▶ **Retrieving & Reusing Configurations:** Management systems which have the capability of retrieving configurations from devices can help in reusing the configurations for other. For example, for an IoT system which has multiple devices and requires same configuration for all devices, it is important to ensure that when a new device is added, the same configuration is applied. For such cases, the management system can retrieve the current configuration from a device and apply the same to the new devices.

2.2.2 SNMP

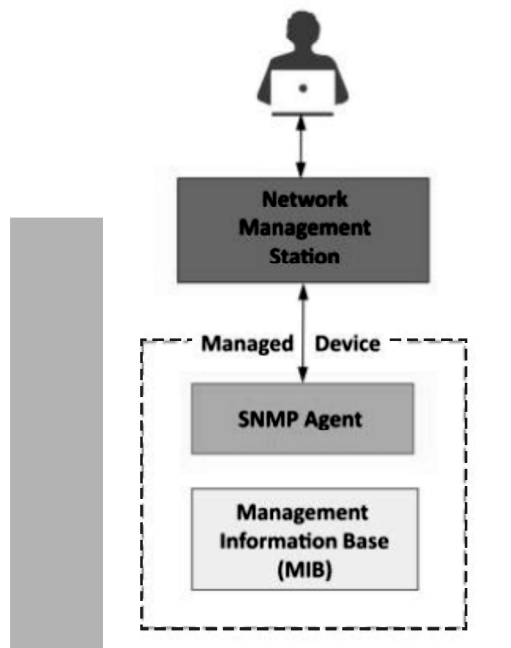
Q7. Write a note on SNMP.

Ans :

SNMP is a well-known and widely used network management protocol that allows monitoring and configuring network devices such as routers, switches, servers, printers, etc.

SNMP component include :

- ▶ Network Management Station (NMS)
- ▶ Managed Device
- ▶ Management Information Base (MIB)
- ▶ SNMP Agent that runs on the device.



Limitations of SNMP

- ▶ SNMP is stateless in nature and each SNMP request contains all the information to process the request. The application needs to be intelligent to manage the device.
- ▶ SNMP is a connectionless protocol which uses UDP as the transport protocol, making it unreliable as there was no support for acknowledgement of requests.
- ▶ MIBs often lack writable objects without which device configuration is not possible using SNMP.
- ▶ It is difficult to differentiate between configuration and state data in MIBs.
- ▶ Retrieving the current configuration from a device can be difficult with SNMP.
- ▶ Earlier versions of SNMP did not have strong security features.

2.2.3 Network Operator Requirements

Q8. Explain about network operator requirements.

Ans :

The following points provide a brief overview of the operator requirements.

- ▶ **Ease of use:** From the operators point of view, ease of use is the key requirement for any network management technology.
- ▶ **Distinction between configuration and state data:** Configuration data is the set of writable data that is required to transform the system from its initial state to its current state. State data is the data which is not configurable. State data includes operational data which is collected by the system at runtime and statistical data which describes the system performance.
- ▶ **Fetch configuration and state data separately :** In addition to making a clear distinction between configuration and state data. It should be possible to fetch the configuration and state data separately from the managed device.
- ▶ **Configuration of the network as a whole:** It should be possible for operators to configure the network as a whole rather than individual devices. This is important for systems which have multiple devices and configuring them within one network wide transaction is required to ensure the correct operation of the system.
- ▶ **Configuration transactions across devices :** Configuration transactions across multiple devices should be supported.
- ▶ **Configuration deltas:** It should be possible to generate the operations necessary for going from one Configurations to another. The devices should support Configuration deltas with minimal state changes.
- ▶ **Dump and restore Configuration:** It should be possible to dump configurations from devices and restore configurations to devices.
- ▶ **Configuration validation :** it should be possible to validate Configurations.

- ▶ **Configuration database schemas:** There is a need for standardized configuration database schemas or data models across Operators.
- ▶ **Comparing configurations:** Devices should not arbitrarily reorder data, so that it is possible to use text processing tools such as diff to compare configurations.
- ▶ **Role-based access control:** Devices should support role-based access control model. so that a user is given the minimum access necessary to perform a required task.
- ▶ **Consistency of access control lists:** It should be possible to do consistency checks of access control lists across devices.
- ▶ **Multiple configurations sets:** There should be support for multiple configurations sets on devices. This way a distinction can be provided between candidate and active configurations.
- ▶ **Support for both data-oriented and task-oriented access control:** While SNMP access control is data-oriented, CLI access control is usually task oriented. There should be support for both types of access control.

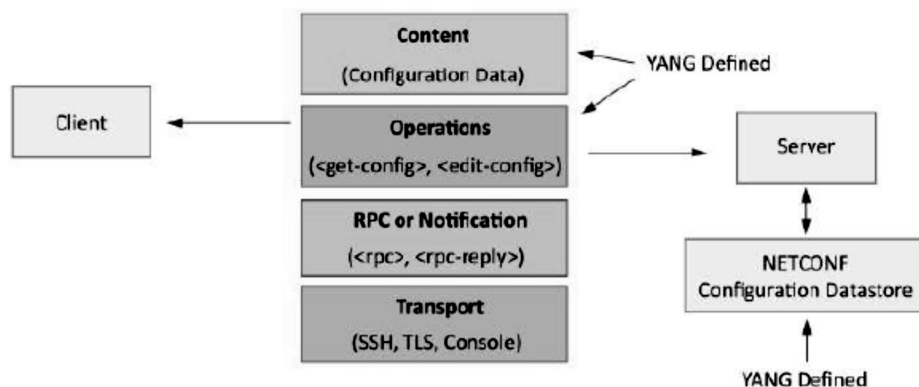
2.2.4 NETCONF

Q9. Write a note on NETCONF protocol.

Ans :

Network Configuration Protocol (NETCONF) is a session-based network management protocol.

NETCONF allows retrieving state or configuration data and manipulating configuration data on network devices.



- ▶ NETCONF works on SSH transport protocol.
- ▶ Transport layer provides end-to-end connectivity and ensure reliable delivery of messages.
- ▶ NETCONF uses XML-encoded Remote Procedure Calls (RPCs) for framing request and response messages.
- ▶ The RPC layer provides mechanism for encoding of RPC calls and notifications.
- ▶ NETCONF provides various operations to retrieve and edit configuration data from network devices.
- ▶ The Content Layer consists of configuration and state data which is XML-encoded.

- ▶ The schema of the configuration and state data is defined in a data modeling language called YANG.
- ▶ NETCONF provides a clear separation of the configuration and state data.
- ▶ The configuration data resides within a NETCONF configuration data store on the server.

2.2.5 Yang

Q10. Write a short note on YANG.

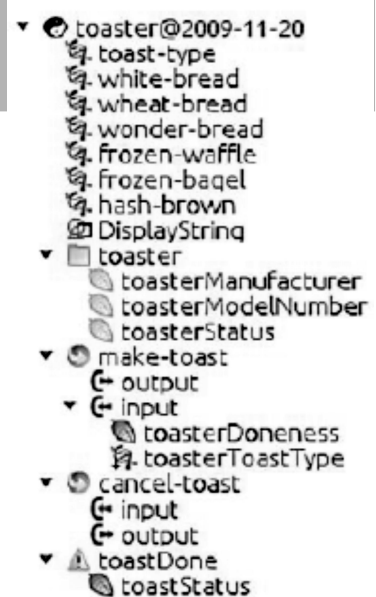
Ans :

YANG is a data modelling language used to model configuration and state data manipulated by the NETCONF protocol.

- ▶ YANG modules contain the definitions of the configuration data, state data, RPC calls that can be issued and the format of the notifications.
- ▶ YANG modules defines the data exchanged between the NETCONF client and server.
- ▶ A module comprises of a number of 'leaf' nodes which are organized into a hierarchical tree structure.
- ▶ The 'leaf' nodes are specified using the 'leaf' or 'leaf-list' constructs.
- ▶ Leaf nodes are organized using 'container' or 'list' constructs.
- ▶ A YANG module can import definitions from other modules.
- ▶ Constraints can be defined on the data nodes, e.g. allowed values.
- ▶ YANG can model both configuration data and state data using the 'config' statement.

YANG Module Example

- ▶ This YANG module is a YANG version of the toaster MIB.
- ▶ The toaster YANG module begins with the header information followed by identity declarations which define various bread types.
- ▶ The leaf nodes ('toasterManufacturer', 'toaster Model Number' and oasterStatus') are defined in the 'toaster' container.
- ▶ Each leaf node definition has a type and optionally a description and default value.
- ▶ The module has two RPC definitions ('make-toast' and 'cancel-toast')

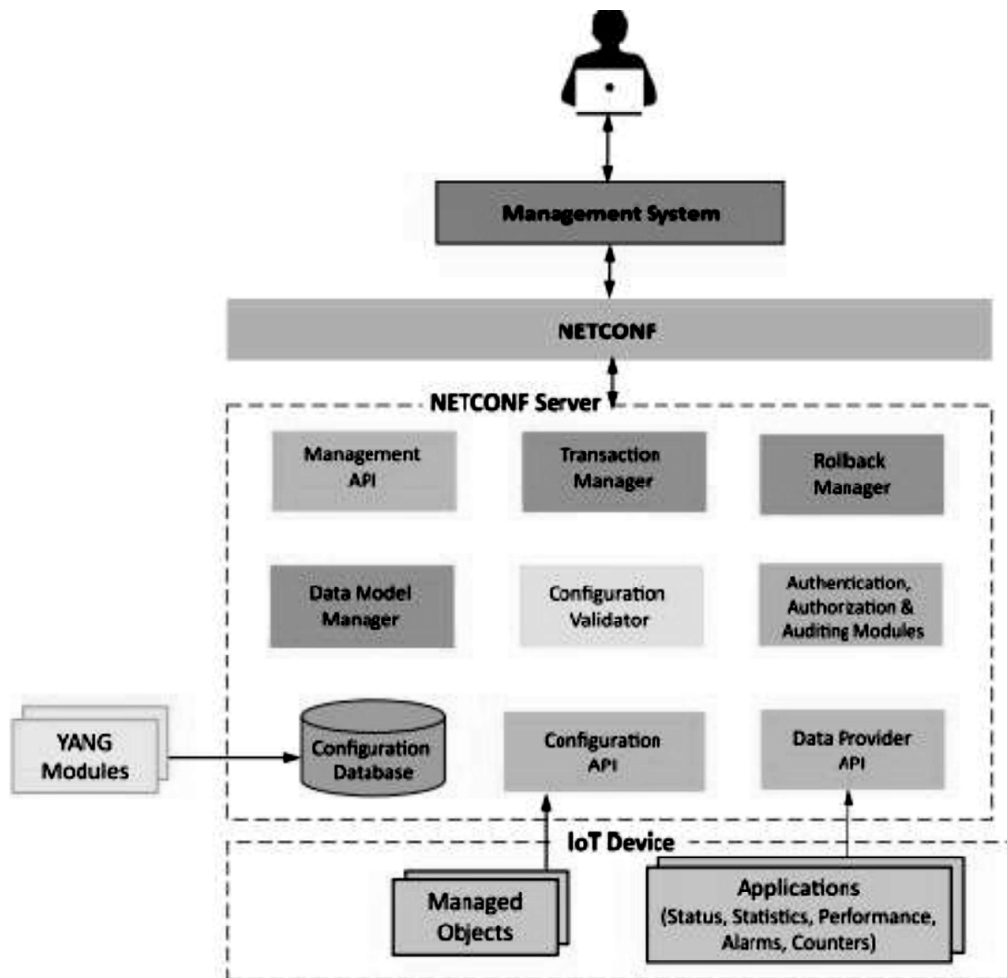


2.2.6 IOT Systems Management w With Netconf-yang

Q11. Explain about IOT systems management with NETCONF-YANG.

Ans :

Following Figure shows the generic approach of IoT device management with NETCONF- YANG.



Let us look at the roles of the various components :

- ▶ **Management System:** The operator uses a Management System to send NETCONF messages to configure the IoT device and receives state information and notifications from the device as NETCONF messages.
- ▶ **Management API:** Management API allows Management applications to start
- ▶ NETCONF sessions, read and write configuration data. read state data, retrieve configurations, and invoke RPCs, programmatically, in the same way as an operator call.

- ▶ **Transaction Manager:** Transaction Manager executes all the NETCONF transactions and ensures that the ACID (Atomicity, Consistency, Isolation, Durability) properties hold true for the transactions.
- ▶ **Rollback Manager:** Rollback manager is responsible for generating all the transactions necessary to rollback a current configuration to its original state.
- ▶ **Data Model Manager:** The Data Model manager keeps track of all the YANG data models and the corresponding managed objects. The Data Model manager also keeps track of the applications which provide data for each part of a data model.
- ▶ **Configuration Validator:** Configuration validator checks if the resulting configuration after applying a transaction would be a valid configuration.
- ▶ **Configuration Database:** The database contains both the configuration and operational data.
- ▶ **Configuration API:** Using the Configuration API the applications on the IoT device can read Configuration data from the configuration data store and write operational data to the operational data store.
- ▶ **Data Provider API:** Applications on the IoT device can register for callbacks for various events using the Data Provider API. Through the Data Provider API. The applications can report statistics and operational data.

2.3 IOT PLATFORMS DESIGN METHODOLOGY

2.3.1 Introduction

Q12. Write a note on IOT platforms design methodology.

Ans :

- ▶ IoT systems comprise of multiple components and deployment tiers.
- ▶ Designing IoT systems can be a complex and challenging task as these systems involve interactions between various components such as
- ▶ IoT devices and network resources, web services, analytics components, application and database servers. Due to a wide range of choices available for each of these components, IoT system designers may find it difficult to evaluate the available alternatives. IoT system designers often tend to design IoT systems keeping specific products / services in mind.

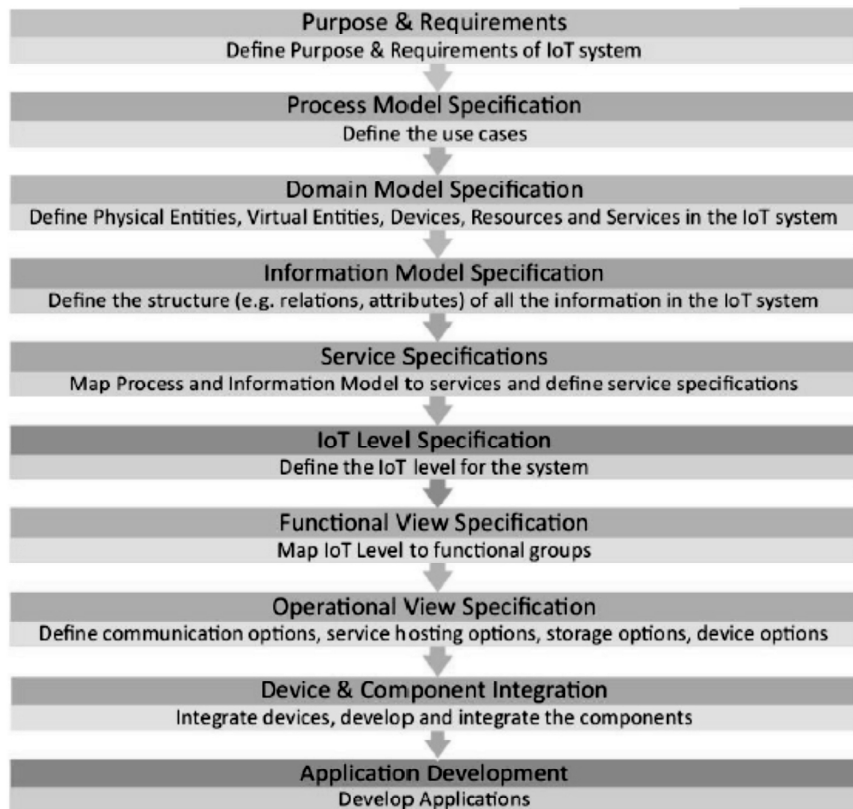
Therefore, these designs are tied to specific product/service choices made. This leads to product, service or vendor lock—in, which while satisfactory to the dominant vendor, is unacceptable to the customer. For such systems, updating the system design to add new features or replacing a particular product/service choice for a component becomes very complex, and in many cases may require complete re—design of the system.

2.3.2 IOT Design Methodology

Q13. Explain about the steps involved in IOT design methodology.

Ans :

IoT Design Methodology – Steps



Step 1: Purpose & Requirements Specification

- ▶ The first step in IoT system design methodology is to define the purpose and requirements of the system. In this step, the system purpose, behavior and requirements (such as data collection requirements, data analysis requirements, system management requirements, data privacy and security requirements, user interface requirements, ...) are captured.

Step 2: Process Specification

- ▶ The second step in the IoT design methodology is to define the process specification. In this step, the use cases of the IoT system are formally described based on and derived from the purpose and requirement specifications.

Step 3: Domain Model Specification

- ▶ The third step in the IoT design methodology is to define the Domain Model. The domain model describes the main concepts, entities and objects in the domain of IoT system to be designed. Domain model defines the attributes of the objects and relationships between objects. Domain model provides an abstract representation of the concepts, objects and entities in the IoT domain, independent of any specific technology or platform. With the domain model, the IoT system designers can get an understanding of the IoT domain for which the system is to be designed.

Step 4: Information Model Specification

- ▶ The fourth step in the IoT design methodology is to define the Information Model. Information Model defines the structure of all the information in the IoT system, for example, attributes of Virtual Entities, relations, etc. Information model does not describe the specifics of how the information is represented or stored. To define the information model, we first list the Virtual Entities defined in the Domain Model. Information model adds more details to the Virtual Entities by defining their attributes and relations.

Step 5: Service Specifications

- ▶ The fifth step in the IoT design methodology is to define the service specifications. Service specifications define the services in the IoT system, service types, service inputs/output, service endpoints, service schedules, service preconditions and service effects.

Step 6: IoT Level Specification

- ▶ The sixth step in the IoT design methodology is to define the IoT level for the system. In Chapter-1, we defined five IoT deployment levels.

Step 7: Functional View Specification

- ▶ The seventh step in the IoT design methodology is to define the Functional View. The Functional View (FV) defines the functions of the IoT systems grouped into various Functional Groups (FGs). Each Functional Group either provides functionalities for interacting with instances of concepts defined in the Domain Model or provides information related to these concepts.

Step 8: Operational View Specification

- ▶ The eighth step in the IoT design methodology is to define the Operational View Specifications. In this step, various options pertaining to the IoT system deployment and operation are defined, such as, service hosting options, storage options, device options, application hosting options, etc.

Step 9: Device & Component Integration

- ▶ The ninth step in the IoT design methodology is the integration of the devices and components.

Step 10: Application Development

- ▶ The final step in the IoT design methodology is to develop the IoT application.

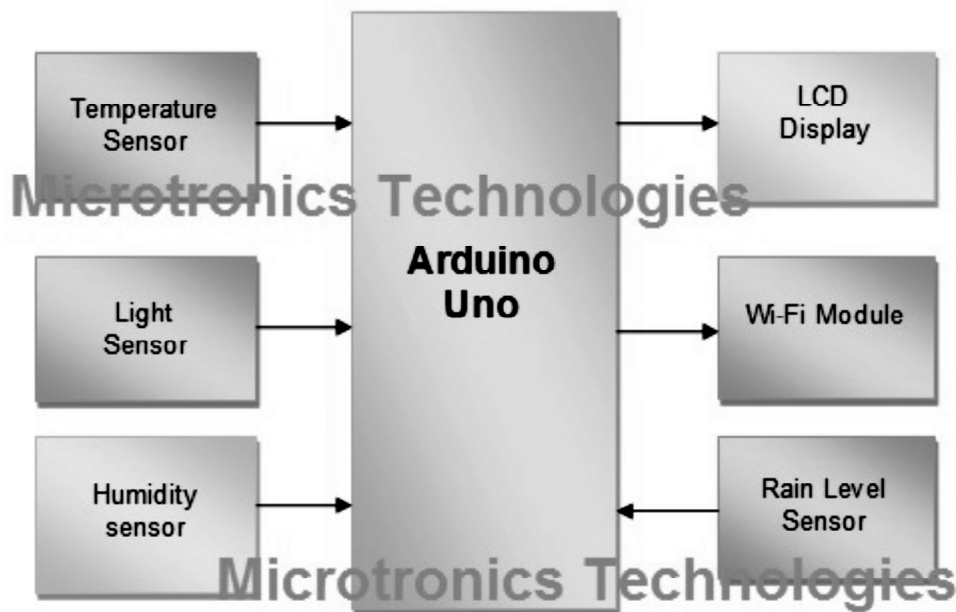
2.3.3 Case Study on IOT System for Weather Monitoring

Q14. Write about the case study on weather monitoring system using IoT.

Ans :

The purpose of weather monitoring system is to collect data on environmental conditions such as temperature, rain level sensor, humidity, and light in an area using multiple end nodes.

The case study connects and stores the data on a web server. Thus user gets Live reporting of weather conditions. Internet connectivity or Internet connection with Wi-Fi is compulsory in this IOT weather monitoring project.

Block Diagram of IOT based weather monitoring system :

Imagine a situation where scientists/nature analysts want to monitor changes in a particular environment say volcano or a rain-forest. And these people are from different places in the world. In this case, SMS based weather monitoring has its own limitations. Since it sends SMS to few numbers. And time for sending SMS increases as the number of mobile numbers increases. In order to send this data to everyone, a person who receives this SMS can upload/add data to some place where everyone can see it. And what else apart from the Internet connects everyone in this world? However, a person doing it manually is time consuming and tedious job. And then there arises a need of an automated solution for this. So in such scenarios, IOT – Internet of Things proves really effective.

Using Internet of Things, we can upload these weather parameters data to the cloud using internet connectivity over a WiFi module through wireless communication. Thus this project is also categorized under Wireless communication projects.

Two things are necessary to view this weather reporting over the Internet. One is the Internet and another is a device to access a URL / website. This device can be laptop or desktop or a tablet or even a smartphone. NOTE that Internet connectivity is required at both places. One where is project is placed and another from where user monitors this data.

Applications of IOT Weather Monitoring System

- 1) IOT weather reporting system has application to farmers as well. The weather forecasting plays very important role in the field of agriculture.
- 2) IOT weather monitoring project proves really helpful for monitoring weather at places like volcano, rain forests. It is quite difficult for a human being to stay for longer time at such places. Or even areas which are exposed to radioactive leakage.

Advantages

- 1) IOT weather monitoring system project using Arduino is fully automated. It does not require any human attention.
- 2) You can get a prior alert of the weather conditions. Suppose you are planning to visit a place and you want to know the weather parameters over that place, then you can just visit a website IOT portal. Future enhancements to this IOT project.

2.3.4 Motivation f For Using Python

Q15. What is the motivation for using Python in IoT ?

Ans :

IoT occupies a place of importance in Wireless Sensor Networks, Data Analytics, Cyber Physical Systems, Big Data and Machine Learning. It is also very focused on real time analytics and processes. So, for the development of an IoT solution, one would need a programming language which spans these diverse fields, while being lightweight and scalable at the same time.

Over the past decade, the popularity of Python as a mainstream programming language has exploded. Notable advantages of Python over other languages include, but are not limited to;

1. It is a very simple language to learn and easy to implement and deploy, so you don't need to spend a lot of time learning lots of formatting standards and compiling options.
2. It is portable, expandable and embeddable, so, it is not system dependent, and hence supports a lot of single board computers on the market these days, irrespective of architecture and operating system.
3. Most importantly, it has a huge community which provides a lot of support and libraries for the language.

2.3.5 Python Packages for IOT

Q16. Write about the Python packages used for IOT.

Ans :

The Python packages used for developing IoT applications.

► **mraa**

mraa is a skeleton GPIO library for most SBCs which support Python. The good thing about it is that there's just one library for all devices, so I don't have to use different ones for an Edison and a Pi. Being a high level library, reading from and writing to pins is a one line affair, and the library also

provides support for communication protocols such as I2C, UART and SPI.

► **sockets**

sockets is a package which facilitates networking over TCP/IP and UDP using Python. It provides access to Berkeley socket APIs to access the Internet. Both TCP/IP and UDP being Transport layer protocols, are ideal for communication with devices on the same WiFi network. One of the more interesting uses of sockets, in my experience is that one can build their own communication protocol using this package as the base.

► **mysqldb**

A database is a no-brainer when it comes to most IoT applications. For something whose sole purpose is to send data to the internet, there should be a database, at least a remote one which stores all this generated data. MySQL is the go-to relational database for most developers. In this regard, mysqldb is a very convenient little tool which circumvents the need to execute shell commands within a Python script to read and write to a database.

► **numpy**

Having used MatLab extensively during my undergraduate studies, I've grown accustomed to dealing with arrays. Python, on the other hand, deals with lists as a substitute for the array which is the same as having a birch tree replace your Rottweiler as the guardian of the house. It just doesn't work. Thankfully, numpy is there to help you out. It is, in essence, a package for scientific computing using Python, very similar to MatLab, but much lighter. The feature I use most is to read sensor data in bulk from my databases and work on them using the inbuilt functions.

► **matplotlib**

Data visualization is one of the most fundamental operations that can be performed. It looks pretty impressive when you convert a huge list of numbers to a concise graph which can be understood intuitively. It's also very useful if you happen to be an academician. You know how important those graphs can be in a publication. matplotlib provides a number of different styles of graphs that can be plotted using local data.

► **pandas**

Another library for data scientists, pandas is a package dedicated towards data analysis. It is in essence, a local alternative to using SQL databases which is more suited to dealing with data as it is built on numpy. It has many advantages over the former, such as a more streamlined approach to data handling and analysis, direct operations on local datasets and the ability to handle heterogeneous and unordered data.

► **opencv**

The big brother of signal processing, image processing, was traditionally the domain of high performance, custom built hardware. Although such devices still carry out the job much faster than their single board counterparts, it is at the very least, a possibility. And, in situations where mobility and connectivity are prioritized over speed, this may just be the solution for those rare times. Opencv is a Python port of the very successful C library for image processing. It contains high-level variants of familiar image processing functions which make photo analysis much easier.

► **tkinter**

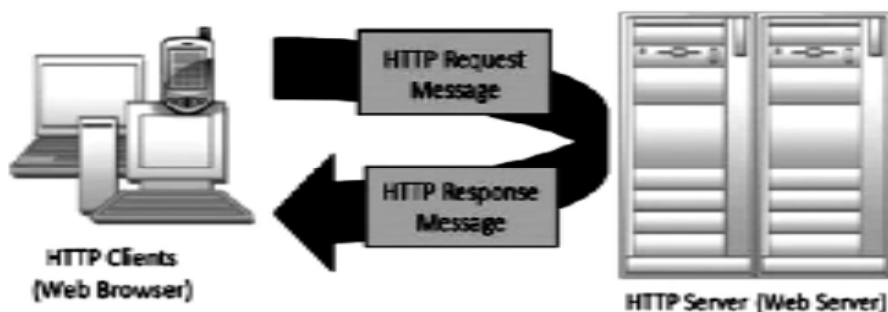
Although this library does come preinstalled with all installations of Python, its still worthy of a mention. Tkinter is a GUI development library which comes bundled in with all distributions of Python. For people who are more comfortable with a stab wound rather than object oriented programming, learning how to use this package may be a bit daunting at first, but the rewards more than make up for the effort. Every aspect of your Python script can be controlled via a completely ad hoc GUI. This is extremely useful in situations such as functionality testing or repeated executions of the same code.

► **tensorflow**

Tensorflow is a package for numerical computations for machine learning. It utilizes a different mathematical representation called data flow graphs which use nodes as mathematical operations and edges as data arrays. This is a very useful library to have if you deal with a lot of non linear datasets or work extensively with decision trees and neural networks.

► **requests**

HTTP is one of the major protocols used in traditional internet based resource exchange, being more suited towards large data exchanges. The requests package is used in Python to make HTTP calls and parse responses. This package is useful when dealing with HTTP based third party cloud services.



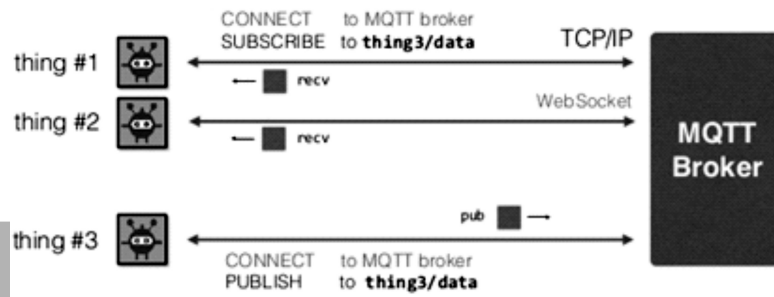
► **paho-mqtt**

MQTT is a protocol developed solely for for the Internet of Things paradigm. Its focus on high speed communication for low payload communication between resource constrained devices. the paho-

mqtt library gives a very user friendly version of the protocol for use with embedded systems. MQTT requests can be made directly within Python, without any additional setup to be done. Especially useful in the prototyping stage.

MQTT

bi-directional, async "push" communication



UNIT III

IoT Physical Devices & Endpoints: What is an IoT Device, Exemplary Device: Raspberry Pi, About the Board, Linux on Raspberry Pi, Raspberry Pi Interfaces, programming Raspberry Pi with Python, Other IoT Devices. IoT Physical Servers & Cloud Offerings: Introduction to Cloud Storage Models & Communication APIs, WAMP AutoBahn for IoT, Xively Cloud for IoT, Python Web Application Framework-Django, Designing a RESTful Web API, Amazon Web Services for IoT, SkyNet IoT Messaging Platform.

3.1 IoT PHYSICAL DEVICES & ENDPOINTS

3.1.1 What is an IoT Device

Q1. What is an IoT device? Write about it.

Ans :

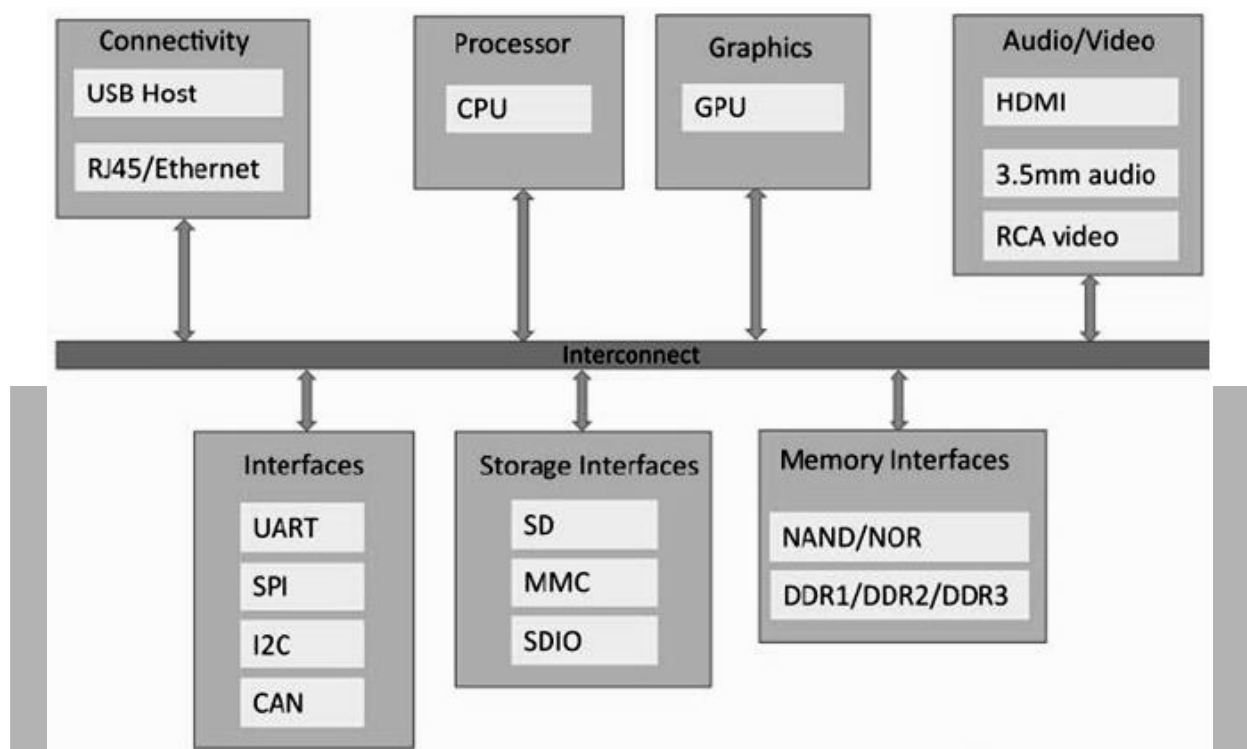
- A "Thing" in Internet of Things (IoT) can be any object that has a unique identifier and which can send/receive data (including user data) over a network (e.g., smart phone, smart TV, computer, refrigerator, car, etc.).
- IoT devices are connected to the Internet and send information about themselves or about their surroundings (e.g. information sensed by the connected sensors) over a network (to other devices or servers/storage) or allow actuation upon the physical entities/environment around them remotely.

IoT Device Examples

- A home automation device that allows remotely monitoring the status of appliances and controlling the appliances.
- An industrial machine which sends information about its operation and health monitoring data to a server.
- A car which sends information about its location to a cloud-based service.
- A wireless-enabled wearable device that measures data about a person such as the number of steps walked and sends the data to a cloud-based service.

Basic building blocks of an IoT Device

- Sensing : Sensors can be either on-board the IoT device or attached to the device.
- Actuation : IoT devices can have various types of actuators attached that allow taking actions upon the physical entities in the vicinity of the device.
- Communication : Communication modules are responsible for sending collected data to other devices or cloud-based servers/storage and receiving data from other devices and commands from remote applications.
- Analysis & Processing : Analysis and processing modules are responsible for making sense of the collected data.

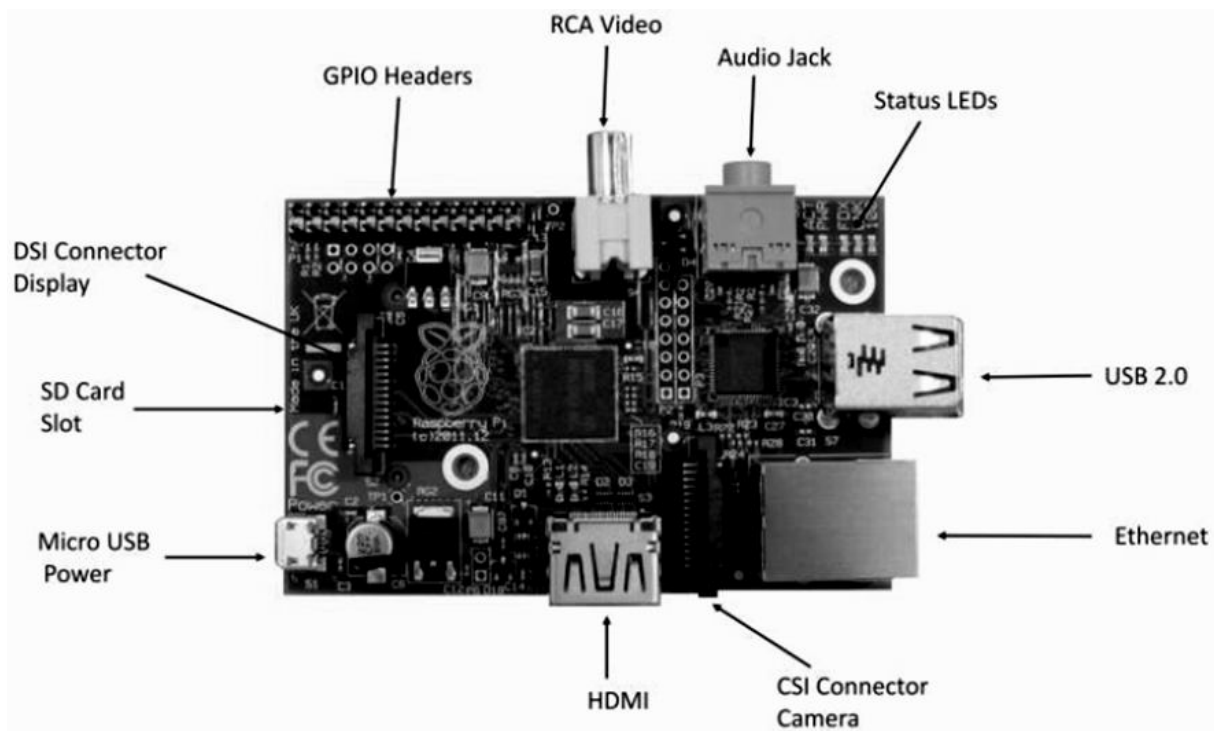
Block diagram of an IoT Device**3.2 EXEMPLARY DEVICE****3.2.1 Raspberry PI****Q2. What is Raspberry PI?***Ans :*

Raspberry Pi is a low-cost mini-computer with the physical size of a credit card.

- Raspberry Pi runs various flavors of Linux and can perform almost all tasks that a normal desktop computer can do.
- Raspberry Pi also allows interfacing sensors and actuators through the general purpose I/O pins.
- Since Raspberry Pi runs Linux operating system, it supports Python "out of the box".

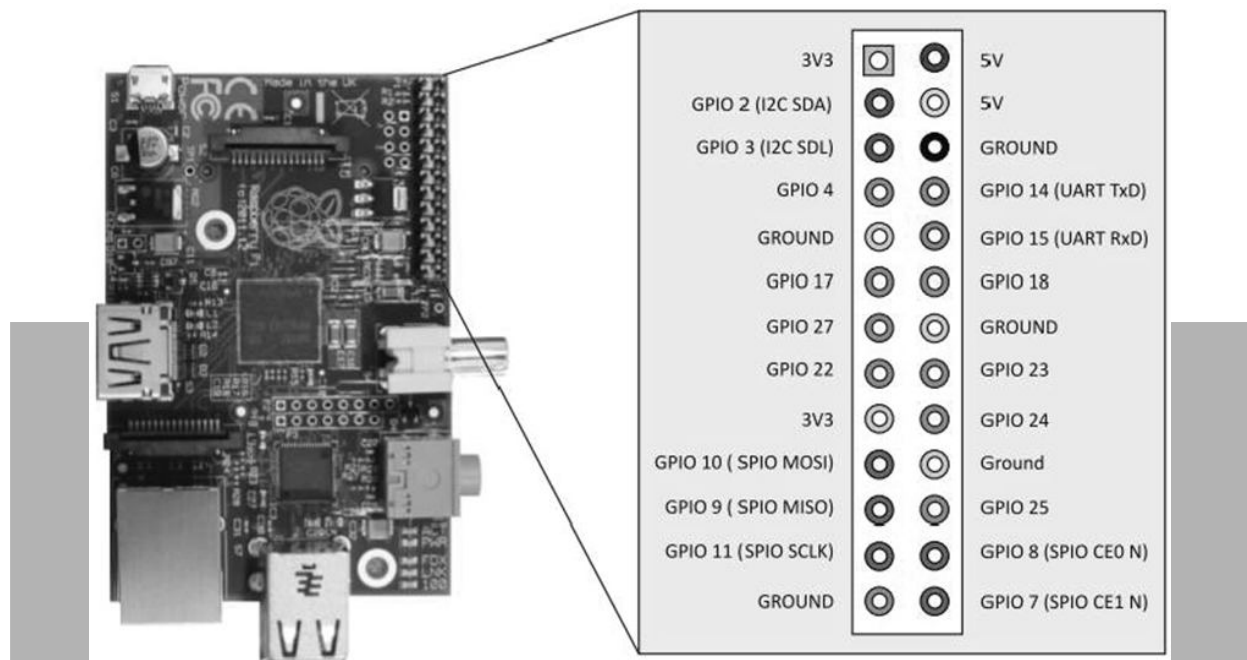
3.2.2 About the Board**Q3. Explain about the structure of a Raspberry Pi.***Ans :*

Processor & RAM: Raspberry Pi is based on an ARM processor. The latest version of Raspberry Pi (Model B, Revision 2) comes with 700 MHz Low Power ARM 1176z-F processor and 512 MB SDR RAM.



- **USB Ports:** Raspberry Pi comes with two USB 2.0 ports. The USB ports on Raspberry Pi can provide a current upto 1000 mA. For connecting devices [that draw current more than 1000mA, an external USB powered hub is required.
- **Ethernet Ports :** Raspberry Pi comes with a standard RJ45 Ethernet port. You can connect an Ethernet cable or a USB Wifi adapter to provide Internet connectivity.
- **HDMI Output :** The HDMI port on Raspberry Pi provides both video and audio output. You can connect the Raspberry Pi to a monitor using an HDMI cable. For monitors that have a DVI port but no HDMI port, you can use an HDMI to DVI adapter/cable.
- **Composite Video Output :** Raspberry Pi comes with a composite video output with an RCA jack that supports both PAL and NTSC video output. The RCA jack can be used to connect old televisions that have an RCA input only.
- **Audio Output :** Raspberry Pi has a 3.5mm audio output jack. This audio jack is used for providing audio output to old televisions along with the RCA jack for video. The audio quality from this jack is inferior to the HDMI output.
- **GPIO Pins :** Raspberry Pi comes with a number of general purpose input/output pins. Figure shows the Raspberry Pi GPIO headers. There are four types of pins on Raspberry Pi - true GPIO pins, I2C interface pins, SPI interface pins and serial Rx and Tx pins.
- **Display Serial Interface (DSI) :** The DSI interface can be used to connect an LCD panel to Raspberry Pi,
- **Camera Serial Interface CSI) :** The CSI interface can be used to connect a camera module to Raspberry Pi.
- **Status LEDs :** Raspberry Pi has five status LEDs.

- **SD Card Slot** : Raspberry Pi does not have a built in operating system and storage. YOU can plug-in an SD card loaded with a Linux image to the SD card slot.
- **Power Input** : Raspberry Pi has a micro-USB connector for power input.



3.2.3 Linux on Raspberry Pi

Q4. What are the various flavors of Linux on Raspberry Pi

Ans :

Raspberry Pi supports various flavors of Linux including:

- **Raspbian** : • Raspbian Linux is a Debian Wheezy port optimized for Raspberry Pi.
- **Arch** : Arch is an Arch Linux port for AMD devices.
- **Pidora** : Pidora Linux is a Fedora Linux optimized for Raspberry Pi.
- **RaspBMC** : RaspBMC is an XBMC media-center distribution for Raspberry Pi.
- **OpenELEC** : • OpenELEC is a fast and user-friendly XBMC media-center distribution.
- **RISC OS** : RISC OS is a very fast and compact operating system.

Q5. Explain how to install Operating system images on Linux using Raspberry Pi.

Ans :

Installing Operating System Images on Linux

Discovering the SD Card Mountpoint and Unmounting It

- Run `lsblk` to see which devices are currently connected to your machine.
- If your computer has a slot for SD cards, insert the card. If not, insert the card into an SD card reader, then connect the reader to your computer.

- Run `lsblk` again. The new device that has appeared is your SD card (you can also usually tell from the listed device size). The naming of the device will follow the format described in the next paragraph.
- The left column of the results from the `lsblk` command gives the device name of your SD card and the names of any partitions on it (usually only one, but there may be several if the card was previously used). It will be listed as something like `/dev/mmcblk0` or `/dev/sdX` (with partition names `/dev/mmcblk0p1` or `/dev/sdX1` respectively), where `X` is a lower-case letter indicating the device (eg. `/dev/sdb1`). The right column shows where the partitions have been mounted (if they haven't been, it will be blank).
- If any partitions on the SD card have been mounted, unmount them all with `umount`, for example `umount /dev/sdX1` (replace `sdx1` with your SD card's device name, and change the number for any other partitions).

Copying the Image to the SD Card

- In a terminal window, write the image to the card with the command below, making sure you replace the input file `if=` argument with the path to your `.img` file, and the `/dev/sdX` in the output file `of=` argument with the correct device name. Make sure the device name is the name of the whole SD card as described above, not just a partition. For example: `sdd`, not `sdds1` or `sddp1`; `mmcblk0`, not `mmcblk0p1`.

`ddbs=4M if=2018-04-18-raspbian-stretch.img of=/dev/sdX conv=fsync`
- Please note that block size set to `4M` will work most of the time. If not, try `1M`, although this will take considerably longer.
- Also note that if you are not logged in as `root` you will need to prefix this with `sudo`.

Copying a Zipped Image to the SD Card

In Linux it is possible to combine the `unzip` and SD copying process into one command, which avoids any issues that might occur when the unzipped image is larger than 4GB. This can happen

on certain filesystems that do not support files larger than 4GB (e.g. FAT), although it should be noted that most Linux installations do not use FAT and therefore do not have this limitation.

The following command unzips the zip file (replace `2018-04-18-raspbian-stretch.zip` with the appropriate zip filename), and pipes the output directly to the `dd` command. This in turn copies it to the SD card, as described in the previous section.

```
unzip -p 2018-04-18-raspbian-stretch.zip |
sudo dd of=/dev/sdX bs=4M conv=fsync
```

Checking the Image Copy Progress

- By default, the `dd` command does not give any information about its progress, so it may appear to have frozen. It can take more than five minutes to finish writing to the card. If your card reader has an LED, it may blink during the write process.
- To see the progress of the copy operation, you can run the `dd` command with the status option.

```
ddbs=4M if=2018-04-18-raspbian-stretch.img of=/dev/sdX status=progress conv=fsync
```

- If you are using an older version of `dd`, the status option may not be available. You may be able to use the `dcfldd` command instead, which will give a progress report showing how much has been written. Another method is to send a `USR1` signal to `dd`, which will let it print status information. Find out the PID of `dd` by using `pgrep -l dd` or `ps a | grep dd`. Then use `kill -USR1 PID` to send the `USR1` signal to `dd`.

Checking whether the Image was Correctly Written to the SD Card

- After `dd` has finished copying, you can check what has been written to the SD card by `dd`-ing from the card back to another image on your hard disk; truncating the new image to the same size as the original; and then running `diff` (or `md5sum`) on those two images.
- Copy the SD card content to an image on your hard drive. `if` is the input file (i.e. the

SD card device), of is the output file to which the SD card content is to be copied, called from-sd-card.img in this example.

ddbs=4M if=/dev/sdX of=from-sd-card.img

- Truncate the new image to the size of the original image, in case the SD card was larger than the original image. Make sure you replace the input file reference argument with the original image name.

truncate —reference 2018-04-18-raspbian-stretch.img from-sd-card.img

- Compare the two images: diff should report that the files are identical.

diff -s from-sd-card.img 2018-04-18-raspbian-stretch.img

- Run sync. This will ensure the write cache is flushed and that it is safe to unmount your SD card.
- Remove the SD card from the card reader.

3.2.4 Raspberry Pi Interfaces

Q6. Write about Raspberry PI Interfaces.

Ans :

Raspberry Pi has serial, SPI and I2C interfaces for data transfer.

- Serial : The serial interface on Raspberry Pi has receive (Rx) and transmit (TX) pins for communication with serial peripherals.
- SPI : Serial Peripheral Interface (SPI) is a synchronous serial data protocol used for communicating with one or more peripheral devices. In an SPI connection, there is one master device and one or more peripheral devices.

There are five pins on Raspberry Pi for SPI interface:

- MISO (Master In Slave Out: Master line for sending data to the peripherals.
- MOSI (Master Out Slave In: Slave line for sending data to the master.
- SCK(Serial Clock) : Clock generated by master to synchronize data transmission.

- CE0 (Chip Enable 0): to enable or disable devices,

- CE0 (Chip Enable 1): To enable or disable devices.

I2C : the I2C interface pins on Raspberry Pi allow you to connect hardware modules. I2C interface allows synchronous data transfer with just two pins SDA (data line) and SCL(clock line).

3.2.5 Programming Raspberry Pi with Python

Q7. Explain how to control LED with raspberry Pi by using Python programming.

Ans :

Raspberry Pi runs Linux and supports Python out of the box. Therefore, you can run any Python program that runs on a normal computer. However, it is the generalpurpose input/output capability provided by the GPIO pins on Raspberry Pi that makes it useful device for Internet of Things, You can interface a wide variety of sensor anti actuators with Raspberry Pi using the GPIO pins anti the SPI, I2C and serial interfaces. input from the sensors connected to Raspberry Pi can be processed and various actions can be taken, for instance, sending data to a server, sending an email, triggering a relay switch.

Controlling LED with Raspberry Pi

Let us start with a basic example of controlling an LED Raspberry Pi.

You can connect the LED to any other GPIO pin as well.

Second program uses the RPi.GPIO module to control the LED on Raspberry Pi. In this program we set pin 18 direction to output and then write true/false alternatively after a delay of one second.

Switching LED on/off from Raspberry Pi console

```
$echo 16 > /sys/class/gpio/vxport
$cd /sys/class/gpio/gpio8
#Set pin 18 direction to out
$echo out. >direction
```

Turn LED on
echo 1 > value

Turn LED off

Echo 3 > value

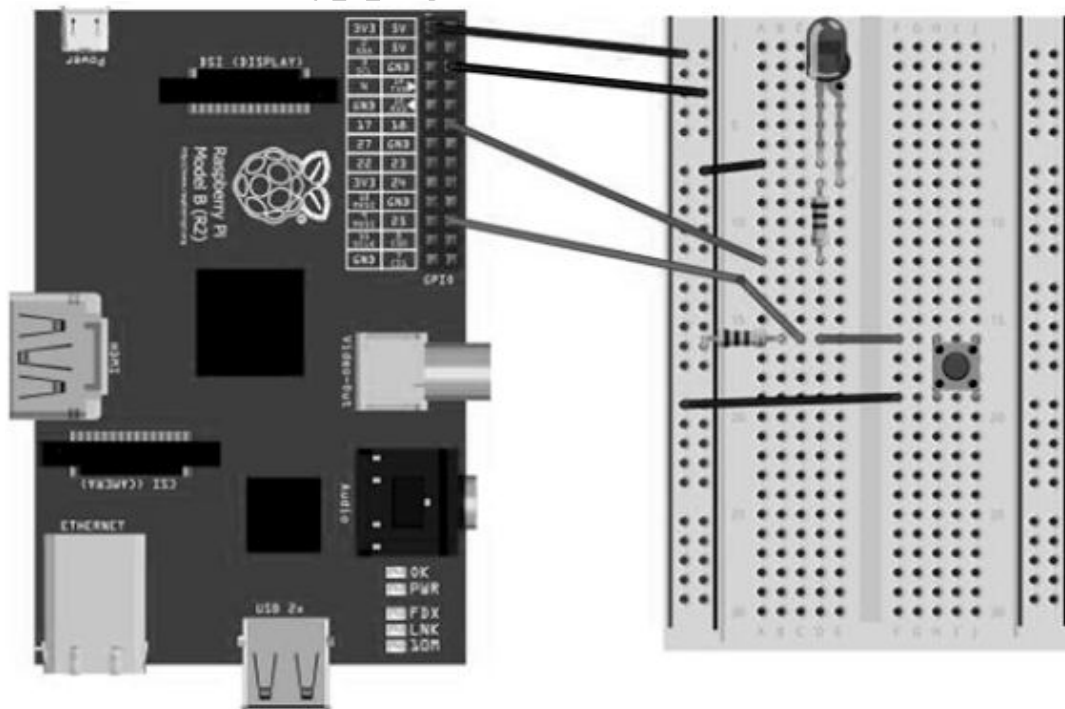
Python program for blinking LED

```
Import RPI.GPIO as GPIO
Import time
GPIO.setmode(l, GPIO.OUT)
while True:
GPIO.output(18,true)
time.sleep(l)
GPIO.output(13, False)
time.sleep(1)
```

Interfacing an LED and Switch with Raspberry Pi

Now let us look at a more detailed example involving in LED and a switch that is used to control the LED.

Figure shows the schematic diagram of connecting an LED and switch to Raspberry Pi. The following program shows a Python program for controlling an LED with a switch. In this example the LED is connected to GPIO pin 18 and switch is connected to pin 25. In the infinite while loop the value of pin 25 is checked and the state of LED is toggled and switch is pressed.



Raspberry Pi Example:

Interfacing LED and switch with Raspberry Pi

```

import sleep
importRPi.GPIO as GPIO
GPIO.setmode(GPIO.BCM)
#Switch Pin
GPIO.setup(25, GPIO.IN)
#LED Pin
GPIO.setup(18, GPIO.OUT)
state=False
deftoggleLED(pin):
state = not state
GPIO.output(pin, state)
while True:
try:
if (GPIO.input(25) == True):
toggleLED(pin)
sleep(.01)
exceptKeyboardInterrupt:
exit()

```

3.2.6 Other IoT Devices

Q8. Write about the other IoT devices which are available.

Ans :

pcDuino

pcDuino is a high performance, cost effective mini PC platform that runs PC like OS such as Ubuntu Linux. It outputs its screen to HDMI enabled TV or monitor via the built in HDMI interface. It is specially targeted for the fast growing demands from the open source community. The platform could run full blown PC like OS with easy to use tool chain and compatible with the popular Arduino ecosystem such as Arduino Shields and open source projects etc. It can also run Android 4.0 ICS.

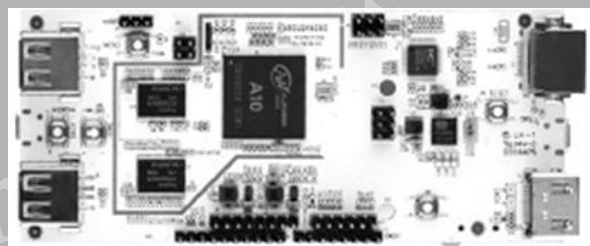


pcDuino targets two markets primarily, i.e., the Raspberry Pi mini PC market and Arduino market as open-source electronics prototyping platform. With pcDuino, user could do lots of fun stuff including but not limited to the follows:

BeagleBone Black

BeagleBone Black is a low-cost, community-supported development platform for developers and hobbyists. Boot Linux in under 10 seconds and get started on development in less than 5 minutes with just a single USB cable.

The BeagleBoard was also designed with open source software development in mind, and as a way of demonstrating the Texas Instrument's OMAP3530 system-on-a-chip.

**Cubieboard**

Cubieboard is a single-board computer, made in Zhuhai, Guangdong, China. The first short run of prototype boards were sold internationally in September 2012, and the production version started to be sold in October 2012.^[1] It can run Android 4 ICS, Ubuntu 12.04 desktop

Fedora 19 ARM Remix desktop, Arch Linux ARM, a Debian-based Cubian distribution or OpenBSD.

It uses the AllWinner A10 SoC, popular on cheap tablets, phones and media PCs. This SoC is used by developers of the limadriver, an open-source driver for the ARM Mali GPU.^[7] It was able, at the 2013 FOSDEM demo, to run iquake 3 at 47 fps in 1024×600.

The Cubieboard team managed to run an Apache Hadoop computer cluster using the Lubuntu Linux distribution.

3.3 IoT PHYSICAL SERVERS & CLOUD OFFERINGS

3.3.1 Introduction to Cloud Storage Models & Communication Apis

Q9. What is cloud storage model?

Ans :

Cloud computing is transformative computing paradigm that involves delivering applications and services over the Internet.

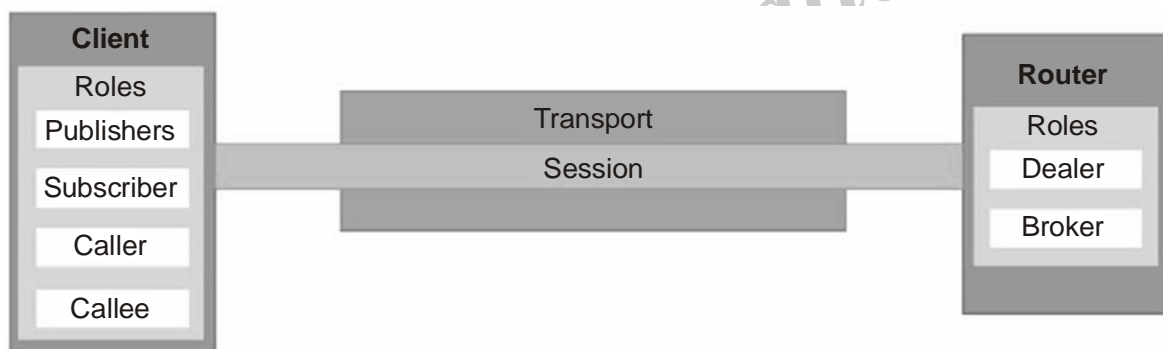
Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction.

3.3.2 WAMP Autobahn for IoT

Q10. Explain about Wireless Application messaging protocol for IoT.

Ans :

- Web Application Messaging Protocol (WAMP) is a sub-protocol of WebSocket which provides publish-subscribe and remote procedure call (RPC) messaging patterns.



- o **Transport:** Transport is channel that connects two peers.
- o **Session:** Session is a conversation between two peers that runs over a transport.
- o **Client:** Clients are peers that can have one or more roles. In publish-subscribe model client can have following roles:
 - **Publisher:** Publisher publishes events (including payload) to the topic maintained by the Broker.
 - **Subscriber:** Subscriber subscribes to the topics and receives the events including the payload.

In RPC model client can have following roles:

- **Caller:** Caller issues calls to the remote procedures along with call arguments.
- **Callee:** Callee executes the procedures to which the calls are issued by the caller and returns the results back to the caller.

- o Router: Routers are peers that perform generic call and event routing. In publish-subscribe model Router has the role of a Broker:
 - Broker: Broker acts as a router and routes messages published to a topic to all subscribers subscribed to the topic.
- In RPC model Router has the role of a Broker:
 - Dealer: Dealer acts as a router and routes RPC calls from the Caller to the Callee and routes results from Callee to Caller.
- o Application Code: Application code runs on the Clients (Publisher, Subscriber, Callee or Caller).

3.3.3 Xively Cloud for IoT

Q11. Write about Xively cloud.

Ans :

Xively is an IoT cloud platform that is “an enterprise platform for building, managing, and deriving business value from connected products”. Moreover, it provides a cloud-based API with an SDK that simplifies the development process. It supports several platforms and technologies:

- Android
- Arduino
- Arm mbed
- C
- Java

and much more. Please, refer to their library website to find more information. Xively is a **Paas** (Platform As a Service) that exposes its services via RESTful APIs.

Moreover, Xively supports messaging service based on MQTT protocol. If you want to have more information about this IoT platform read Xively Architecture. In this IoT project tutorial, we will use the free Xively account.

Now we know a bit more about Xively and the services it provides, it is useful to start the IoT project. As an example, we will build a simple system that monitors the plant status health using a set of sensors that help us to measure environment conditions and the soil moisture.

Xively uses some basic concepts:

Xively device: It is your “device” or the IoT project you are building. Consider it as an envelope containing the data.

Channels: It is the streaming coming from your sensors. Usually, the channel number is equal to the sensor number (as long as you want to monitor them all).

So the first step is adding the “device” to our account:

Device Name

Smart Plant

Device Description optional

Plant Health monitor

Privacy You own your data, we help you share it. more info

☐ Private Device
You use API keys to choose if and how you share a device's data.

☒ Public Device
You agree to share a device's data under the CC0 1.0 Universal license. The Device's data is indexed by major search engines, and its Feed page is publicly viewable.

Then we define the channels. In this example we have 4 channels as it is clear in the picture below:

Add Channel ID required

e.g. sensor1

Tags Use a comma to separate tags. e.g. energy, project:name=my_pr

Units e.g. Watts

Symbol e.g. W

Current Value

Save Channel Cancel

That's all. Now you have to use the Xively library for your device and start sending data to the cloud.

3.3.4 Python Web Application Framework-DJANGO

Q12. Write a note on DJANGO – A python Web application framework.

Ans :

- Django is an open source web application framework for developing web applications in Python.
- A web application framework in general is a collection of solutions, packages and best practices that allows development of web applications and dynamic websites.
- Django is based on the Model-Template-View architecture and provides a separation of the data model from the business rules and the user interface.
- Django provides a unified API to a database backend.
- Thus web applications built with Django can work with different databases without requiring any code changes.
- With this flexibility in web application design combined with the powerful capabilities of the Python language and the Python ecosystem, Django is best suited for cloud applications.
- Django consists of an object-relational mapper, a web templating system and a regular-expression based URL dispatcher.

Django Architecture

Django is Model-Template-View (MTV) framework.

Model

- The model acts as a definition of some stored data and handles the interactions with the database. In a web application, the data can be stored in a relational database, non-relational database, an XML file, etc.

A Django model is a Python class that outlines the variables and methods for a particular type of data.

Template

- In a typical Django web application, the template is simply an HTML page with a few extra placeholders. Django's template language can be used to create various forms of text files (XML, email, CSS, Javascript, CSV, etc.)

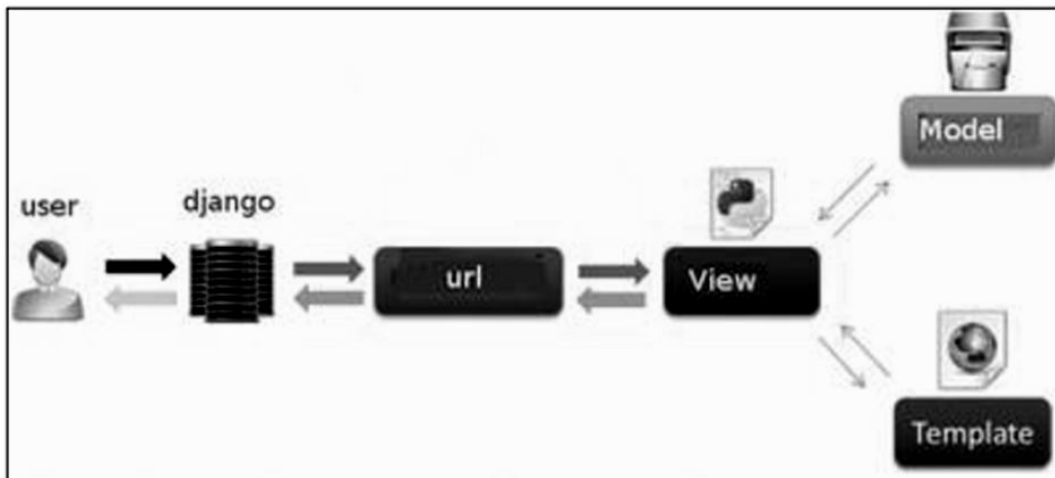
View

- The view ties the model to the template. The view is where you write the code that actually generates the web pages.

View determines what data is to be displayed, retrieves the data from the database and passes the data to the template.

The Model-View-Template (MVT) is slightly different from MVC. In fact the main difference between the two patterns is that Django itself takes care of the Controller part (Software Code that controls the interactions between the Model and View), leaving us with the template. The template is a HTML file mixed with Django Template Language (DTL).

The following diagram illustrates how each of the components of the MVT pattern interacts with each other to serve a user request-



The developer provides the Model, the view and the template then just maps it to a URL and Django does the magic to serve it to the user.

3.3.5 Designing a Restful Web API

Q13. Explain how to design a RESTful Web API.

Ans :

One of the most difficult decisions to make when designing a RESTful API is to find the fine line between making your API simple enough that it can be tested directly in the browser. Devices should instead use published APIs to store and retrieve data.

There are some points need to be considered while designing the APIs for IoT:

- API should be RESTful in nature. We should use GET, PUT, POST and DELETE. Respectively, these should be used to retrieve, update, create and delete data.
- For public data, no authorization is needed but for everything else an API key should serve for authorization.
- Since there is likely to be lots of devices sending data, there should be parameters to search and filter. In addition, it should be possible to define and operate on meaningful groups of devices.
- For every data stream, there should be a unique URI.
- Data should be presented in standardized formats such as JSON or XML. We should avoid sending binary data into the cloud, the format of which is probably known only to sensor devices.

When designing a REST system, the state of the distributed application must be assigned to the different components.

Session state encompasses the control flow and the interactions between the components. Following the statelessness constraint, the session state must be kept only on clients. On the one hand, this makes requests a bit more verbose since every request must contain all the information necessary to process it. On the other hand, this makes servers efficient, since they do not have to keep any state about their clients. Requests can easily be distributed over multiple worker threads or server instances. For the IoT systems, it lowers the memory requirements for server implementations, which is particularly important for constrained servers and servers serving large amount of clients.

Resource state includes the more persistent data of an application. This can be static data such as device descriptions, persistent data such as system configuration, but also dynamic data such as the current value of a sensor on a thing.

Resource Modelling

Important part of RESTful API design is to model the system as a set of resources whose state can be retrieved and/or modified and where resources can be potentially also created and/or deleted.

Resource representations have a media type that tells how the representation should be interpreted. Typical media types for IoT systems include "text/plain" for simple UTF-8 text, "application/octet-stream" for arbitrary binary data, "application/json" for JSON "application/senml+json" Sensor Markup Language (SenML) formatted data, "application/cbor" for CBOR "application/exi" for EXI.

Uniform Resource Identifiers (URIs)

Uniform Resource Identifiers (URIs) are used to interact with a resource, to reference a resource from another resource, to advertise or bookmark a resource, or to index a resource by search engines.

For RESTful IoT applications, typical schemes include "https",

"coaps", "http", and "coap". These refer to HTTP and CoAP, with and without Transport Layer Security (TLS) (CoAP uses Datagram TLS (DTLS) the variant of TLS for UDP.)

These four schemes also provide means for locating the resource; using the HTTP protocol for "http" and "https", and with the CoAP protocol for "coap" and "coaps". If the scheme is different for two URIs (e.g., "coap" vs. "coaps"), it is important to note that even if the rest of the URI is identical, these are two different resources, in two distinct namespaces.

HTTP/CoAP Methods

GET

The GET method requests a current representation for the target resource. Only the origin server needs to know how each of its resource

identifiers corresponds to an implementation and how each implementation manages to select and send a current representation of the target resource in a response to GET.

A payload within a GET request message has no defined semantics.

A response to a successful GET request is cacheable; a cache may use it to satisfy future, equivalent GET requests. The GET method is safe and idempotent.

POST

The POST method requests that the target resource process the representation enclosed in the request according to the resource's own specific semantics.

If one or more resources has been created on the origin server as a result of successfully processing a POST request, the origin server sends a 201 (Created) response containing a Location header field that provides an identifier for the resource created and a representation that describes the status of the request while referring to the new resource(s).

The POST method is not safe nor idempotent.

PUT

The PUT method requests that the state of the target resource be created or replaced with the state defined by the representation enclosed in the request message payload. A successful PUT of a given representation would suggest that a subsequent GET on that same target resource will result in an equivalent representation being sent.

The PUT method is not safe, but is idempotent.

DELETE

The DELETE method requests that the origin server remove the association between the target resource and its current functionality.

If the target resource has one or more current representations, they might or might not be destroyed by the origin server, and the associated storage might or might not be reclaimed, depending entirely on the nature of the resource and its implementation by the origin server.

The DELETE method is not safe, but is idempotent.

HTTP/CoAP Status/Response Codes

The status codes consist of three digits (e.g., "404" or "4.04") where the first digit expresses the class of the code. Implementations do not need to understand all status codes, but the class of the code must be understood. Codes starting with 1 are informational; the request was received and being processed. Codes starting with 2 indicate successful request. Codes starting with 3 indicate redirection; further action is needed to complete the request. Codes starting with 4 and 5 indicate errors. The codes starting with 4 mean client error (e.g., bad syntax in request) whereas codes starting with 5 mean server error; there was no apparent problem with the request but server was not able to fulfil the request.

Responses may be stored in a cache to satisfy future, equivalent requests. HTTP and CoAP use two different patterns to decide what responses are cacheable. In HTTP, the cacheability of a response depends on the request method. In CoAP, the cacheability of a response depends on the response code.

Security Considerations

This document does not define new functionality and therefore does not introduce new security concerns. However, security consideration from related specifications apply to RESTful IoT design. These include:

- HTTP security
- CoAP security
- URI security.

3.3.6 Amazon Web Services for IoT

Q14. What Is Amazon EC2?

Ans :

Amazon Elastic Compute Cloud (Amazon EC2) provides scalable computing capacity in the Amazon Web Services (AWS) cloud. Using Amazon EC2 eliminates your need to invest in hardware up front, so you can develop and deploy applications faster. You can use Amazon EC2 to launch as many or as few virtual servers as you need, configure

security and networking, and manage storage. Amazon EC2 enables you to scale up or down to handle changes in requirements or spikes in popularity, reducing your need to forecast traffic.

For more information about cloud computing, see [What is Cloud Computing?](#)

Features of Amazon EC2

Amazon EC2 provides the following features:

- Virtual computing environments, known as *instances*
- Preconfigured templates for your instances, known as *Amazon Machine Images (AMIs)*, that package the bits you need for your server (including the operating system and additional software)
- Various configurations of CPU, memory, storage, and networking capacity for your instances, known as *instance types*
- Secure login information for your instances using *key pairs* (AWS stores the public key, and you store the private key in a secure place)
- Storage volumes for temporary data that's deleted when you stop or terminate your instance, known as *instance store volumes*
- Persistent storage volumes for your data using Amazon Elastic Block Store (Amazon EBS), known as *Amazon EBS volumes*
- Multiple physical locations for your resources, such as instances and Amazon EBS volumes, known as *regions* and *Availability Zones*
- A firewall that enables you to specify the protocols, ports, and source IP ranges that can reach your instances using *security groups*
- Static IPv4 addresses for dynamic cloud computing, known as *Elastic IP addresses*
- Metadata, known as *tags*, that you can create and assign to your Amazon EC2 resources
- Virtual networks you can create that are logically isolated from the rest of the AWS cloud, and that you can optionally connect to your own network, known as *virtual private clouds (VPCs)*.

Q15. What is AWS Auto Scaling ?*Ans :***.AWS Auto Scaling**

AWS Auto Scaling monitors your applications and automatically adjusts capacity to maintain steady, predictable performance at the lowest possible cost. Using AWS Auto Scaling, you can setup scaling for multiple resources across multiple services in minutes. AWS Auto Scaling provides a simple, powerful user interface that lets you build scaling plans for Amazon EC2 instances and Spot Fleets, Amazon ECS tasks, Amazon DynamoDB tables, and Amazon Aurora Replicas.

AWS Auto Scaling makes scaling simple with recommendations that allow you to optimize performance, costs, or balance between them. If you're already using Amazon EC2 Auto Scaling, you can now combine it with AWS Auto Scaling to scale additional resources for other AWS services. With AWS Auto Scaling, your applications always have the right resources at the right time.

There is no additional charge for AWS Auto Scaling. You pay only for the AWS resources needed to run your applications and Amazon CloudWatch monitoring fees. To get started you can use the AWS Management Console, Command Line Interface (CLI), or SDK.

Q16. What i Amazon Simple Storage Service (Amazon S3)*Ans :***Amazon Simple Storage Service (Amazon S3)**

Amazon Simple Storage Service (Amazon S3) is a scalable, high-speed, low-cost, web-based cloud storage service designed for online backup and archiving of data and application programs. S3 was designed with a minimal feature set and created to make web-scale computing easier for developers.

Q17. What Is Amazon Relational Database Service (Amazon RDS)?*Ans :*

Amazon Relational Database Service (Amazon RDS) is a web service that makes it easier to set up, operate, and scale a relational database

in the cloud. It provides cost-efficient, resizable capacity for an industry-standard relational database and manages common database administration tasks.

Q18. Amazon DynamoDB?*Ans :*

Amazon DynamoDB is a fully managed NoSQL database service that provides fast and predictable performance with seamless scalability. DynamoDB lets you offload the administrative burdens of operating and scaling a distributed database, so that you don't have to worry about hardware provisioning, setup and configuration, replication, software patching, or cluster scaling. Also, DynamoDB offers encryption at rest, which eliminates the operational burden and complexity involved in protecting sensitive data.

Q19. What Is Amazon Kinesis Data Streams?*Ans :*

Use Amazon Kinesis Data Streams to collect and process large streams of data records in real time.

You'll create data-processing applications, known as *Amazon Kinesis Data Streams applications*. A typical Amazon Kinesis Data Streams application reads data from a *Kinesis data stream* as data records. These applications can use the Kinesis Client Library, and they can run on Amazon EC2 instances. The processed records can be sent to dashboards, used to generate alerts, dynamically change pricing and advertising strategies, or send data to a variety of other AWS services. For information about Kinesis Data Streams features and pricing,

Q20. What is Amazon Simple Queue Service?*Ans :*

Amazon Simple Queue Service (Amazon SQS) offers a secure, durable, and available hosted queue that lets you integrate and decouple distributed software systems and components. Amazon SQS offers common constructs such as dead-letter queues and cost allocation tags. It provides a generic web services API and it can be accessed by any programming language that the AWS SDK supports.

3.3.7 Skynet IoT Messaging Platform

Q21. Write about SKYNET IoT Messaging Platform.

Ans :

SkyNet is a machine to machine instant messaging platform for the Internet of Things. You can easily connect your devices to your private SkyNet running on your own hardware or you can connect to the public SkyNet.

SkyNet offers a realtimewebsocket API as well as a Node.js NPM module to make event-driven IoT development fast and easy. When nodes and devices register with SkyNet, they are assigned a unique id known as a UUID along with a security token. (Note: tokens can be provided by the request rather than having SkyNet assign one for you.)

Upon connecting your node or device to SkyNet, you can query and update devices on the network and send machine-to-machine (M2M) messages in an RPC-style fashion.

But most sensors or controllers don't speak JSON. The great thing is that SkyNet is the ideal middleware that facilitates talking to these devices. You can connect your controller to an Arduino or a Raspberry and let those do the translation from and to your device. So you can turn out your lights through Twitter and receive text messages from your thermostat.

But it goes further, SkyNet allows you to query the connected devices. You can find all drones in your area, or all weather stations in the area you are planning to visit. It is a feature request that you can query a node to retrieve the services it can deliver.

The **SkyNet** tool offers a complete suite of patch management tools in one application. Kiss goodbye Windows updates, stop worrying about security on each and every device you manage – and start relaxing while **SkyNet** takes care of the work for you, ensuring *all* essential system updates are applied.

- Supports **ALL** updates (including non-security updates) for Microsoft Windows and other Microsoft products including Office and Exchange
- Covers all 5 major browsers – Explorer, Chrome, Firefox, Safari and Opera
- Support includes most commonly exploited Adobe and Oracle Java
- Other vendor support including Apple, Mozilla, Zip tools and more, as well as Instant Messaging Clients such as Skype and Yahoo Messenger

UNIT IV

Case Studies of IoT Design: Home Automation, Cities, Environment, Agriculture, Productivity Applications. Introduction to Data Analytics for IoT, Apache Hadoop, YARN, Oozie, Spark, Storm, Health Monitoring Case study. An IoT Tool: chef, Chef Case Studies.

4.1 CASE STUDIES OF IoT DESIGN

4.1.1 Home Automation and Cities

Q1. Explain about the IoT case study on Home automation.

Ans :

Home automation has three major parts:

- Hardware
- Software/Apps
- Communication protocols

Each of these parts is equally important in building a truly smart home experience for your customers. Having the right hardware enables the ability to develop your IoT prototype iteratively and respond to technology pivots with ease.

A protocol selected with the right testing and careful consideration helps you avoid performance bottlenecks that otherwise would restrict the technology and device integration capabilities with sensors and IoT gateways.

Another important consideration is the firmware that resides in your hardware managing your data, managing data transfer, firmware OTA updates, and performing other critical operations to make things talk.

Applications of Home Automation

Rebuilding consumer expectations, home automation has been projected to target wide array applications for the new digital consumer. Some of the areas where consumers can expect to see home automation led IoT-enabled connectivity are:

- Lighting control
- HVAC
- Lawn/Gardening management
- Smart Home Appliances
- Improved Home safety and security
- Home air quality and water quality monitoring
- Natural Language-based voice assistants
- Better Infotainment delivery
- AI-driven digital experiences
- Smart Switches
- Smart Locks
- Smart Energy Meters

The list is still not exhaustive and will evolve over the time to accommodate new IoT use cases.

Now that you are familiar with home automation applications, let's have a detailed look at what components are involved in building a typical home automation prototype.

Home Automation Components

We have talked about them before, but let's clearly separate our components that will finally help you build a realistic model of what major components are involved in building a smart home. The major components can be broken into:

- IoT sensors
- IoT gateways
- IoT protocols

- IoT firmware
- IoT cloud and databases
- IoT middleware (if required)

IoT sensors involved in home automation are in thousands, and there are hundreds of home automation gateways as well. Most of the firmware is either written in C, Python, Node.js, or any other programming language.

The biggest players in IoT cloud can be divided into a platform-as-a-service (PaaS) and infrastructure-as-a-service (IaaS).

Home Automation Sensors

There are probably thousands of such sensors out there that can be a part of this list, but since this is an introduction towards smart home technology, we will keep it brief. We will break down IoT sensors for home automation by their sensing capabilities:

- Temperature sensors
- Lux sensors
- Water level sensors
- Air composition sensors
- Video cameras for surveillance
- Voice/Sound sensors
- Pressure sensors
- Humidity sensors
- Accelerometers
- Infrared sensors
- Vibrations sensors
- Ultrasonic sensors

Depending upon what you need, you may use one or many of these to build a truly smart home IoT product. Let's have a look at some of the most commonly used home automation sensors.

Home Automation Protocols

One of the most important parts of building a home automation product is to think about protocols — protocols that your device will use to communicate to gateways, servers, and sensors. A few years ago, the only way to do so was by either using Bluetooth, Wi-Fi, or GSM. But due to added expenses on cellular SIM cards and low performance of Wi-Fi, most such solutions didn't work.

Bluetooth survived and later evolved as Bluetooth Smart or Bluetooth Low Energy. This helped bring a lot of connectivity in the "mobile server powered economy." Essentially, your phone would act as a middleware to fetch data from BLE-powered sensors and send it over to the internet.

When looking at the major home automation protocols, the following top the list:

- Bluetooth Low Energy or Bluetooth Smart: Wireless protocol with mesh capabilities, security, data encryption algorithms, and much more. Ideal for IoT-based products for smart homes.
- Zigbee: Low cost, mesh networked, and low power radio frequency-based protocol for IoT. Different Zigbee versions don't talk to each other.
- X10: A legacy protocol that utilizes powerline wiring for signaling and control.
- Insteon: Communicates with devices both wirelessly and with wires.
- Z-wave: Specializes in home automation with an emphasis on security.
- Wi-Fi: Needs no explanation.
- UPB: Uses existing power lines installed in a home. Reduces costs.
- Thread: A royalty-free protocol for smart home automation, uses a 6lowpan.
- ANT: An ultra low-power protocol helping developers build low-powered sensors with a mesh distribution capabilities.
- 6lowpan

Home Automation Architecture

This architecture supports the following considerations for home automation solutions:

- End to end security mechanisms involving multilevel authentication
- End to end data encryption, including the link layer
- Flexible and configurable access and authorization control
- Powerful cloud infrastructure

- Network agnostic with built-in feedback loops
- Configurable cloud-based rules engine
- API endpoints
- Data scalability
- NoSQL databases

Home Automation Gateways

For developing a home automation product, often a standalone product sending data to a server is not enough. Due to battery and protocol limitations, the data from a sensor or sensors present in a home has been routed through an IoT gateway.

To select the perfect gateway for your IoT home automation, consider some of these factors:

- Communication protocols supported
- Real-time capabilities
- MQTT, CoAP, and HTTPS support
- Security and configuration
- Modularity

When it comes to building IoT gateways, modularity and hybrid IoT protocol support top the list when a product is in the early stages of market introduction.

To incorporate a gateway in your home automation stack, you can consider the following options:

- Either create a gateway from the ground up using existing hardware stacks for prototyping (using Raspberry Pi, Intel Edison, etc). Then, when a PoC is validated, you can create your own custom hardware.
- Or, you can use existing gateway modules like Ingics BLE gateway. These gateways are extremely easy to customize and connect with your cloud services and devices. However, they may or may not offer the same level of support that you need to build certain features.

For example, a gateway with a bad networking queue may result in traffic congestion, or it may not support the required protocols that you wish to use.

Further, pivoting with these gateways to some other technology stack may become very difficult. It should be emphasized that they are extremely good for robust prototyping needs.

Home Automation Programming Languages

The following programming languages dominate the home automation space: Python, Embedded C, C, Shell, Go, and JavaScript (Node.js). This has mainly happened due to the sheer optimization of the languages for similar use cases.

Home Automation Frameworks

Everyone, from high-growth startups to billion-dollar consumer-focused enterprises, is now using the help of home automation frameworks to build connected products to delight consumers.

There are more than 15 different smart home frameworks available for IoT developers to use and build their next generation of connected home products. Some of these frameworks are open source and some are closed-source.

CITIES

Q2. Write a Case study on Smart Parking.

Ans :

Smart Parking

Finding a parking space during rush hours in crowded cities can be time consuming and frustrating. Furthermore, drivers blindly searching for parking spaces create additional traffic congestion.

Smart parking make the search for parking space easier and convenient for drivers. Smart parking are powered by IoT systems that detect the number of empty parking slots and send the information over the Internet to smart parking application hack-ends.

These applications can be accessed by the drivers from smart-phones. tablets and in-car navigation systems.

In smart parking, sensors are used (for each parking slot, to detect whether the slot is empty or occupied. This information is aggregated by a local controller and then sent over the Internet to the database.

Here is the high level architecture for the smart parking system along with a mathematical model. The parking system that we propose comprises of various actors that work in sync with one another. Below is the mathematical model that defines our smart parking system.

Symbol	Meaning
T	Parking time
C	Driver's car number
P	Amount paid
U	User ID
S	Parking slot
M_i	Driver
O	Occupancy rate
$X()$	Input function
$Y()$	Output function
$F()$	Computation function
$I()$	Identity function

Table : Nomenclature Table

$M_i \rightarrow X(T, C, P, U, S)$ // Driver provides input to the input function

$X() \rightarrow F(S, T)$ // Input function notifies the computation function

$X() \rightarrow I(P, C, U)$ // Input function notifies the identity function

$O_i = F(S, T) \rightarrow Y()$ // Computation function notifies the output function and the resultant is stored in form of the occupancy rate.

$O_i = 0 \rightarrow 1$ // Occupancy rate can either be 0 or 1. Where 0 specifies occupied and 1 means vacant.

The following figure gives an outlined view of the complete system.

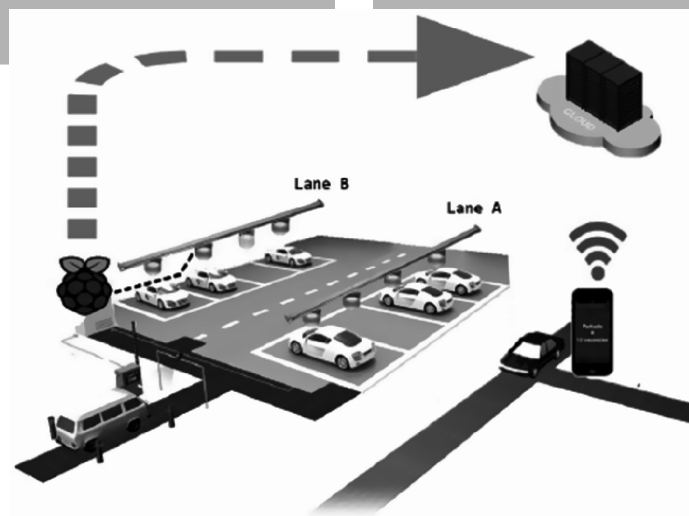


Fig. : Smart Parking System

Talking of the above mentioned figure, it depicts a parking area where our parking system is implementation along with the way in which communication happens between various actors.

The primary actors that constitute the parking system are:

- **Parking Sensors:** For our parking system we have made use of sensors like Infrared, PassiveInfrared(PIR) and Ultrasonic Sensors. The work of these sensors is the same i.e. to sense the parking area and determine whether a parking slot is vacant or not.

In this case we are using ultrasonic sensors to detect the presence of a car. The ultrasonic sensors are wirelessly connected to raspberry pi using the ESP8266 chip. An ESP8266 WiFi chip comprises of a self contained SOC with integrated TCP/IP protocolstack that allows any microcontroller to access a WiFi network. The sensors are connected to a 5V supply either from raspberry pi or an external source. External source being more preferable.

- **Processing Unit:** It comprises of Raspberry pi which is a processor on chip. The processing unit acts like an intermediate between the sensors and cloud. All the sensors are wirelessly connected to the processing unit. A single raspberry pi unit comprises of 26 GPIO pins i.e. 26 different sensors can be connected to it. However we can increase this number by attaching a multiplexer (MUX) to it. It is essential that the ground of raspberry pi and sensors must be connected in order to transfer data using the GPIO pins. There is a python script running on the chip that checks the status of various GPIO pins and updates this information onto the cloud. Data collected from various sensors is sent to the raspberry pi through the esp 8266 chip. The raspberry pi then transmits this data to the IBM MQTT Server through MQTT protocol over a channel. MQTT (Message Queue TelemetryTransport) Protocol is a publish-subscribe based "lightweight" messaging protocol that is used on top of the TCP/IP protocol. It is designed to establish

connections across remote locations where limited amount of data needs to be transferred or in cases of low bandwidth availability.

- **Mobile application:** The mobile application acts like an interface for the end users to interact with the system. The application is developed in Apache Cordova and Angular Js framework using Javascript as a programming language. The purpose of using Apache Cordova is to create applications that can run on both android and iOS platform with the same source code. The application is connected with the IBM MQTT server through a secure channel and a 2factor authorization. The purpose of this mobile application is to provide information regarding availability of parking spaces and allowing the end user to book a slot accordingly. Transfer of data takes place in JSON format between IBM MQTT server and the mobile application. In order to ensure proper communication both the Raspberry pi and mobile application must be subscribed to a particular channel on IBM MQTT server.

- **The Cloud :** The IBM MQTT server is hosted on cloud. Cloud acts as a data base to store all the records related to parking areas and end users that have access to the system. It keeps a track of every user connected to the system and maintains information such as time at which the car was parked, time duration for parking a car, amount paid by the user and mode of payment.. It is due to the flexible nature of cloud which permits the system to add any number of users at any time of the day. Continuous backup is made of the data stored on cloud in order to ensure easy and quick recovery of data in case of any kind of system failure.

On closely looking at the figure one gets to see that empty parking spaces are indicated by red light in Lane A whereas green light in Lane B. This is due to the fact that in case of Lane A although there is no car currently parked but there still is a red light because the slot has already been booked by some user. On the other hand, the parking slot in Lane B shows green light because it neither has a booking nor a car parked in it.

Implementation & Working

The complete process of booking a parking slot, parking a car in that slot and leaving the parking area is done with the help of the following flow chart.

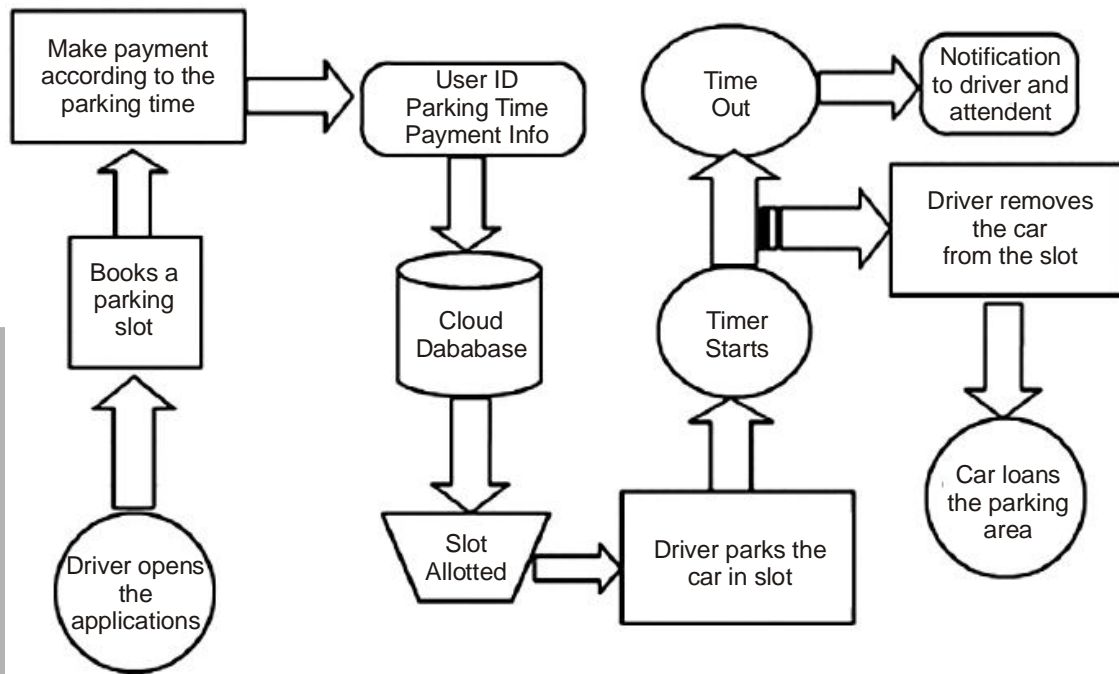


Fig. : Flow chart of the system

Below are the steps that a driver needs to follow in order to park its car using our parking system.

Step 1: Install the smart parking application on your mobile device.

Step 2: With the help of the mobile app search for a parking area on and around your destination.

Step 3: Select a particular parking area.

Step 4: Browse through the various parking slots available in that parking area.

Step 5: Select a particular parking slot.

Step 6: Select the amount of time (in hours) for which you would like to park your car for.

Step 7: Pay the parking charges either with your e-wallet or your credit card.

Step 8: Once you have successfully parked your car in the selected parking slot, confirm your occupancy using the mobile application.

The above mentioned procedure for booking a slot and parking a car. Once the driver has parked its car in the selected slot it needs to confirm its occupancy.

4.1.2 Environment

Q3. Write a Case study on Weather monitoring system.

Ans :

Case study on Weather monitoring system

This IoT based weather monitoring system is developed using powerful development platform Raspberry Pi board. Raspberry Pi board is helpful to minimize the system hardware. So here in this

project use of any external microcontroller, ADC and communication module is avoided. This system uses Temperature and Humidity Sensor (DHT11), Light Intensity Sensor (LDR), Rain Water Level Measuring Sensor developed using marked scale with ULN2803, Pressure and Altitude Sensor (BMP180). All these sensors are interfaced with GPIO header of Raspberry Pi board. To get real time monitoring of data from sensors Ethernet network is used. Block diagram of complete system is as shown in Fig. 1. Interfacing diagram of sensors with Raspberry Pi board is as shown in Fig. 2. Flow chart for system data flow is as shown in Fig. 3.

- Raspberry PI :** IoT based real time weather monitoring system is developed using Raspberry Pi platform. In proposed system architecture ARM11 processor is centre core as shown in figure. In this system single core 32 bit ARM11 processor is used. It is having 512MB RAM. This board has on-board USB ports, Ethernet port, HDMI port, SD card slot. It is easy to give internet connection to this board by using Ethernet port or USB port. Different sensors are interfaced with general purpose input output (GPIO) of Raspberry Pi board for environmental parameter monitoring. Also 5V, 1A power supply is given to the Raspberry Pi board through micro USB slot.

An SD card of 8GB is used to store the operating system as well as all programs and files needed for this project. Keyboard and mouse is connected to the Raspberry Pi board through USB ports. Monitor is connected to the Raspberry Pi board through HDMI port using HDMI to VGA cable. Ethernet port is used to give internet connection to the system via LAN. This board also has a standard 3.5mm mini analog audio jack, intended to drive high impedance loads (like amplified speakers) and a standard RCA-type jack that provides composite NTSC or PAL video signals. Along with all this board is having 15 pin CSI (Camera Serial Interface) connector for camera module interfacing and 15 pin DSI (Display Serial Interface) connector for LED or LCD display.

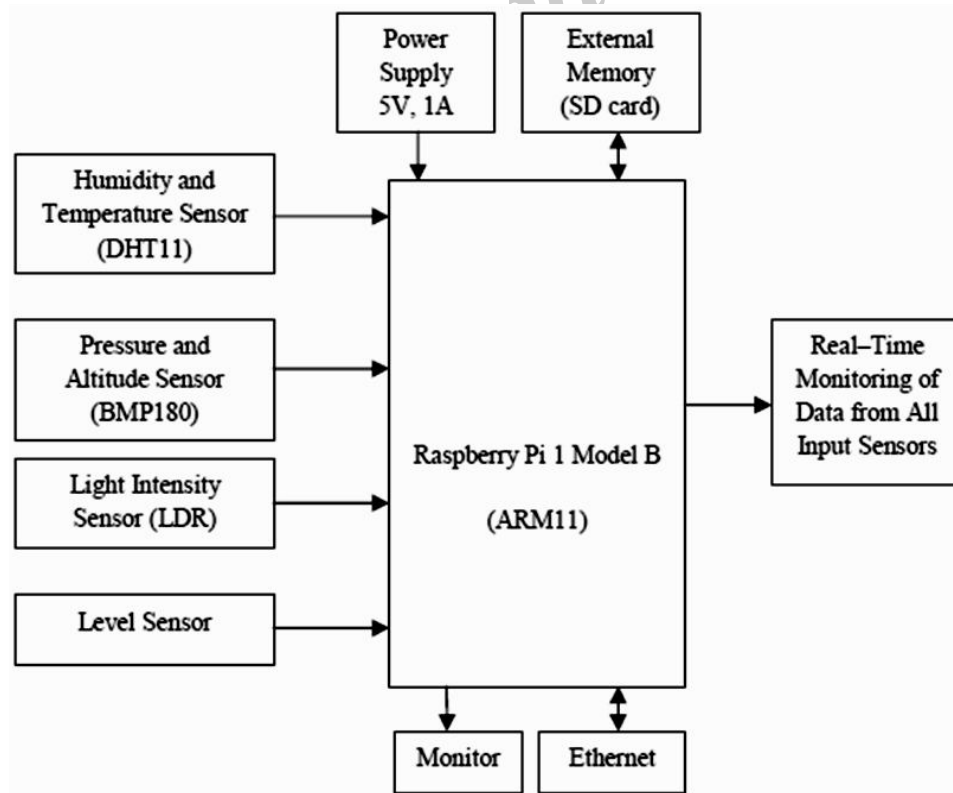


Fig. : Block diagram of complete system

2. **Humidity and Temperature Sensor (DHT11):** GPIO9 of Raspberry Pi is used to interface DHT11 with Raspberry Pi board for humidity (in %) and temperature (in °C) measurement using single wire serial interface (SPI). For humidity measurement resistive type component and for temperature measurement negative temperature coefficient (NTC) component is used by this sensor. Output of DHT11 is calibrated digital signal which Raspberry Pi can understand and no need to have analog to digital converter. DHT11 works on 3-5.5V voltage supply and 0.5-2.5mA current supply.

3. **Pressure and Altitude Sensor (BMP180):** BMP180 is interfaced with Raspberry Pi using I2C interface. SDA and SCL pins are used to interface. This sensor measures atmospheric pressure (in Pa) and altitude from sea level (in m) using piezo-resistive technology. Output from BMP180 is fully calibrated digital signal so there is no need of ADC. It works on 1.8-3.6V voltage and 5mA current supply in standard mode.

4. **Light Intensity Sensor (LDR):** LDR is interfaced at GPIO10 of Raspberry Pi. Light Dependent Resistor (LDR) is used for light intensity measurement. It gives analog output. To avoid the use of ADC, using capacitor circuit is developed. By measuring capacitor charging time which is dependent on LDR resistance light intensity is measured.

With the change in light intensity LDR resistance changes. So at high light intensity resistance is less and at low light intensity resistance is high. Depending on this principle capacitor charging time changes. Depending on capacitor charging time, light intensity is measured as High, Medium and Low. On thingspeak.com it is shown in terms of percentage.

5. **Level Sensor:** For rain water level measurement level measuring scale is designed which is interfaced with ULN2803. ULN2803 is interfaced at GPIO7, GPIO8, GPIO18 and GPIO23 of Raspberry Pi for

IN1, IN2, IN3 and IN4 respectively. On thingspeak.com graphical representation is shown in terms of percentage.

6. **Raspbian:** Raspbian is free and open source software. Raspbian operating system is based on Linux kernel. An SD card is used to install an operating system.

7. **Python:** Python is used for general purpose programming which is free to use and high level language. Python is a interpreted, interactive, object-oriented and beginner's language. Python can runs on Linux kernel. IDLE (Integrated Development and Learning Environment) is the special text editor software used for programming in python.

8. **HTTP:** REST (Representational State Transfer protocol) is a scalable architecture that allows things to communicate over Hyper Text Transfer Protocol and is easily adaptable for IoT applications to provide a communication from things to a central web server. Interoperability between computer systems on the internet is provided by the web services like REST or RESTful. Web service is a service offered via World Wide Web by an electronic device to another electronic device, communicating with each other. In that web technology such as HTTP was originally designed for human to machine communication and now utilized for machine to machine communication. HTTP (Hyper Text Transfer Protocol) is an application protocol used for data communication for the World Wide Web. HTTP is the protocol which is used to exchange or transfer hypertext. HTTP functions as request-response protocol in the client-server computing model. Here client is a web browser and application running on the computer or system is a server. In this system Raspberry Pi itself act as a server. HTTP request is sent by the client to the server. Then response message is sent by the server to the client. In the response completion status of request and requested content is sent.

9. System Flow Chart:

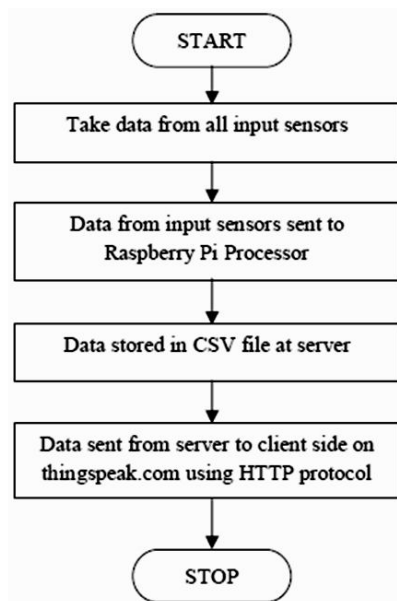


Fig. : System Flow Chart

4.1.3 Agriculture

Q4. Write a case study on smart agriculture.

Ans :

Agriculture is the major source of income for the largest population in India and is major contributor to Indian economy. Although few initiatives have also been taken by the Indian Government for providing online and mobile messaging services to farmers related to agricultural queries, agro vendor's information to farmers. We need a dynamic model which collects real time data, All agriculture entities need to be connected to have decision making system to increase the production and ease the distribution of agricultural products from farmers to marketing agencies and from vendors to farmers. Such system will also be responsible for controlling other parameters like agro product rates.

Smart mobile phones are available now days to many users including in the rural areas. Beagle black bone is a cheap IoT device which can be interfaced to soil and environmental sensors to collect soil properties and current environmental conditions. This motivates to develop a cost effective and portable sensor kit for sensing the soil properties for current requirements of fertilizers. The soil data from farmlands needs to be collected through sensor kit and sent to AgroCloud storage for further

processing. The collected big-data then can be analysed for the required actions for production. few models in agriculture domain using one or more of the technologies mentioned; the dynamic model is needed that provides an integrated approach to:

1. Monitor various soil properties from each farmland and environmental conditions periodically through portable cost effective IoT device and usable by multiple users, enquires about crop production details to the farmers after crop harvesting and stores these details at the central place as in the cloud storage. This in result producing Big-data over the time and will be analysed for fertilizer requirements for current crop, mapping of crop production to soil properties at that time, next crop to be cultivated etc. This will be helpful for increase in production.
2. Connect all agricultural entities together including farmers, agro marketing agencies, agro product vendors and Ministry of agriculture and AgroBanks. This will facilitate distribution of products from farmers to buyers and from agro vendors to farmers. Through the Ministry of agriculture farmers will be able to get notifications about new schemes announced by the government for agriculture sector.

Multidisciplinary Model for Smart Agriculture

The proposed architecture of multidisciplinary model as shown in figure 1 consists of the five modules:

1. SensorKit Module.
2. MobileApp Module.
3. AgroCloud Module.
4. Big-Data Mining, Analysis and Knowledge Building Engine Module.
5. Government & AgroBanks UI

SensorKit module is portable IoT device with soil and environment sensors. MobileApp module provides interface to the users. AgroCloud Module consists of storage, Big-Data mining, analysis and knowledge building engine and application module to communicate with the users. Government and AgroBanks user interface is an web interface for information related to agricultural schemes and loans.

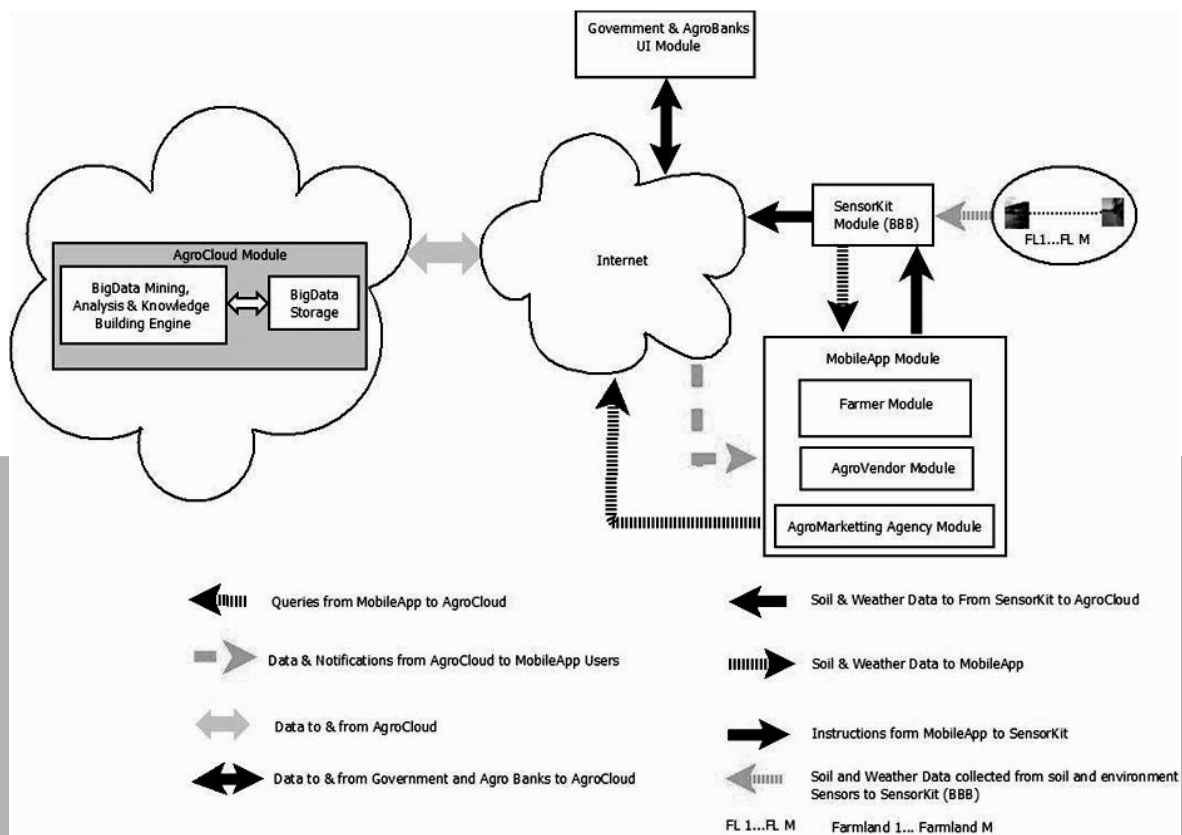


Fig. 1 : Proposed Architecture for Multidisciplinary model for Smart Agriculture

SensorKit Module

This module is an important part of this architecture and is responsible for soil sampling at periodic intervals to get soil property values.

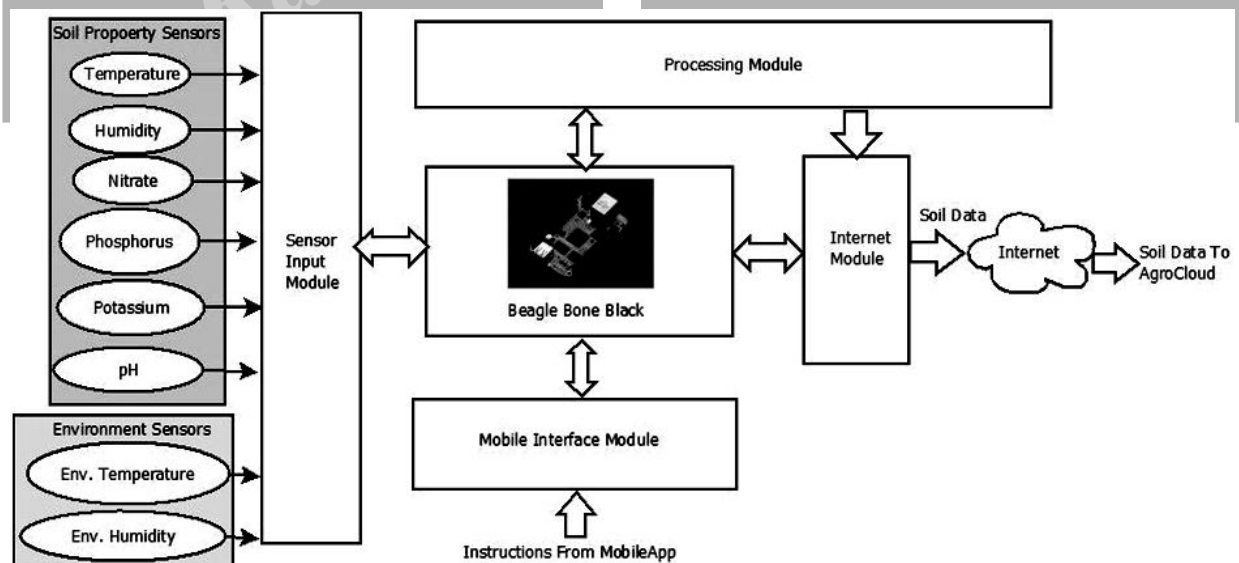


Fig. 2 : SensorKit Module.

Figure 2 shows SensorKit module. SensorKit is a cost effective and portable kit in which we have considered the use of beagle black bone which is IoT enabled device with memory and processing capability, GPS sensor to detect the positional information. The major components of this kit are soil nutrient sensor devices connected to it. Soil attributes sensors we have considered for this model are soil pH sensor, soil moisture sensor, Phosphorus (P), Potassium (K), Nitrate (N) sensors which are interfaced to the IoT device.

MobileApp Module:

Mobile applications need to be installed on the end users mobile phone. It has three parts

- (a) UI for farmer
- (b) UI for agro marketing agency
- (c) UI for agro vendors including fertilizer, pesticide providers and seed providers.

Initially the end user has to register to the mobile app with few credentials including identity information, user type, address, geographical locations and other necessary details. If end user is farmer then has to send few credentials regarding the farmland information consisting of approximate location and total area for each farmland. The soil information per farmland is gathered through SensorKit. SensorKit gets the required instructions from MobileApp. The information will be sent and stored on AgroCloud Big-Data storage. SensorKit also collects and sends the soil information to cloud storage when the crop cultivation is in progress. Through these app farmers get suggestions regarding the fertilizers required and its amount for better crop results and cost savings. This app is also used for sending the notifications to the users. When the crop is harvested, the total production information for each crop will be sent to the cloud storage from the farmer along with current soil characteristics after cultivation of that crop. This information is stored in the cloud storage along with the time-stamp details. Agro marketing agencies responsible for purchasing harvested crops from farmers has to send the periodic updates related to changes in cost and their purchase requirements. Agro product vendors are responsible for selling fertilizer, seed, and pesticide and agricultural equipment's. Agro vendors have to send updates related to products and cost changes periodically. Mobile application module is shown in figure 3.

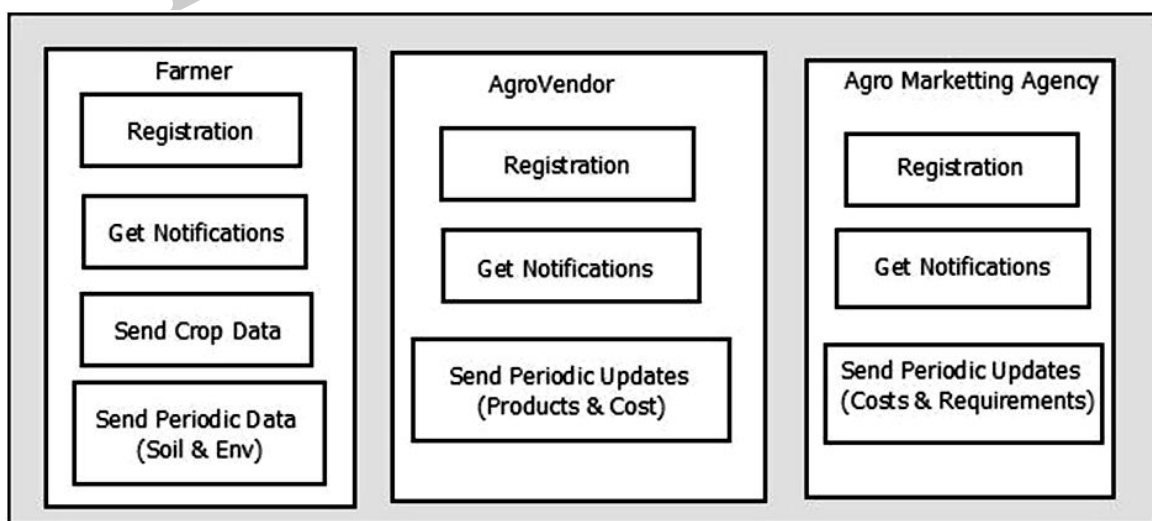


Fig. 3 : MobileApp Module

AgroCloud Module

All the users of agriculture sector needs to be registered to AgroCloud through MobileApp. AgroCloud storage consisting of Big-Data storage will store all the details of farmer, agro marketing agent details, and agro vendors and service providers (fertilizer/pesticide/seed and agro equipment providers) details and government schemes for agriculture sector including bank loans for farmers and concessions given on seed and/or fertilizers.

This module also stores periodic data collected through soil and environment sampling. As larger and larger number of end users gets connected to this service and the data size grows rapidly over the time resulting into the Big-Data. The AgroCloud module with Big-Data storage, Big-Data Mining, Analysis and Knowledge Building Engine and application module is shown in figure 4.

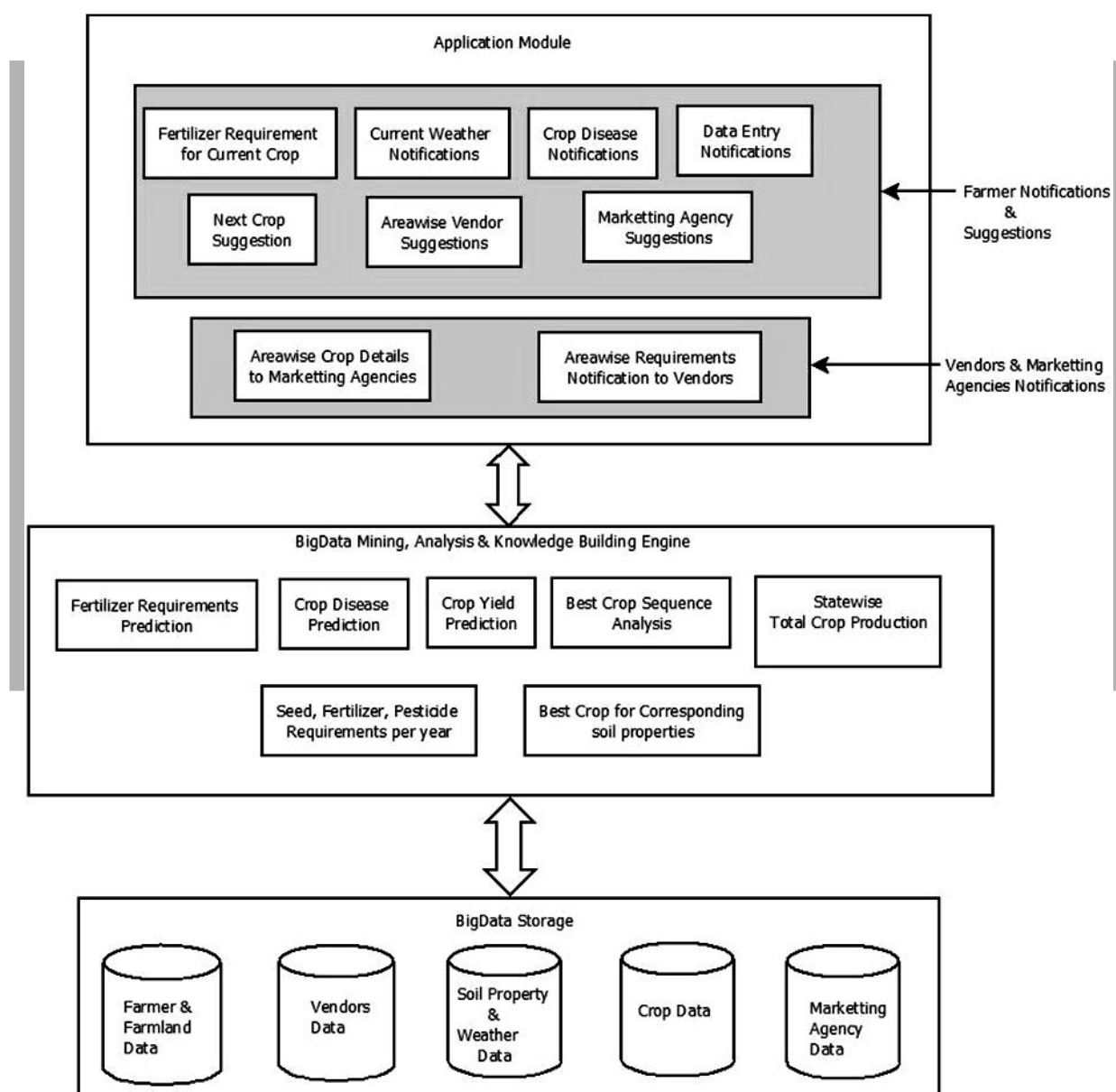


Fig. 4 : AgroCloud Module

This module resides at AgroCloud and as shown in figure 4 plays important role in decision making for the fertilizer requirements for current crop based on current soil properties for better yields, crop disease prediction based on current soil properties and current weather conditions, crop yield prediction, best crop sequence analysis from the data collected over the period, best crop for corresponding soil properties, watering required based on soil moisture level. This database also provides information of region wise crop production details for each crop, total crop production for each crop in the state, based on this and current requirements for the consumers will be helpful to control the costs for each agro product.

As this database collects information over the years for soil properties and crop information details with its production amount for each farmland, inference results with data mining can be calculated for better crop sequences to be carried for best production and to preserve good soil health. As well as this database can provide suggestions to the farmers for crops to be taken on the farmland with peculiar soil properties based on previous stock of agro products and current requirements in the market. Bigdata analysis can be carried out to estimate future production of each product based on previous knowledge base.

4.1.4 Productivity Applications

Q5. Write a note on IoT printer.

Ans :

IoT Printer

The "Internet of Things" is the idea of pervasive connections between physical objects and the online world. These connected devices don't just idly sit around waiting for commands or files...they're active *agents* that anticipate your needs and can push or pull data from the internet.

Our *Internet of Things Printer* is a small, internet-connected thermal printer that can have a daily weather forecast ready before you head out in the morning, a puzzle to work on while riding the subway, provide a list of "tweets" relating to your interests...or any other task you can program!

This second version of the printer is built around the Raspberry Pi, a tiny computer that packs a wallop: more processing power, more RAM and the potent Linux operating system. The new kit has an easier time handling graphics and looks super tidy with its wireless networking:

There are many smart printers available that can actually be managed remotely. Embedded technology in the printers allows for automatic supply replenishment, automated meter readings, and remote diagnostics. This allows businesses to have greater productivity. For example, smart printers can send alerts when maintenance is needed or ink needs to be ordered.

Connected with managed print services

Proactive management and automatic monitoring is at the root of managed print services (MPS). Many enterprises are using some form of MPS to better handle and manage their printer and MFP fleets. However, with no standardization between printer brands, MPS providers have to rely on a combination of tools, or a specialized tool that can track and monitor usage and device status across a varied fleet.

Perhaps print management could be simplified by adding smart sensors that are standardized on all print devices. IoT and Machine to Machine (M2M) technology can change printers for the better, with the print industry having already jumped ahead in regards to connectivity.

Helping businesses with M2M technology

Most printer hardware has cloud connectivity, along with improved document processing abilities and intelligence. With the addition of networking technology and sensors, businesses can receive crucial real-time knowledge about their printer usage.

The need for routine maintenance is lessened by remote monitoring and analytics because any issues can be noticed and corrected before affecting a business negatively. Along with this, device downtime is reduced, due to the timely delivery of printer information. Certain repairs can even be done remotely, allowing individuals to use their time more wisely.

Furthermore, the real-time data can be utilized to become more energy efficient, improving the impact to the environment and lowering carbon emissions. For example, printers can be put into a rest mode, or turned off completely.

The end result of M2M technology is improved efficiency and reduced costs for businesses. New levels of innovation through partnerships between vendors, technology and communications providers are needed to continuously be able to adapt and meet the needs of clients.

IT, mobile environments and enterprise printing are far from the same. As M2M technology grows and becomes more affordable, it will perform a larger part in consumer and business environments alike.

4.1.5 Introduction to Data Analytics for IoT

Q6. Write a note on the relationship between data analytics and IoT.

Ans :

IoT analytics is the application of data analysis tools and procedures to realize value from the huge volumes of data generated by connected Internet of Things devices.

The potential of IoT analytics is often discussed in relation to the Industrial IoT. The IIoT makes it possible for organizations to collect and analyze data from sensors on manufacturing equipment, pipelines, weather stations, smart meters, delivery trucks and other types of machinery. IoT analytics offers similar benefits for the management of data centers and other facilities, as well as retail and healthcare applications.

IoT data can be thought of as a subset and a special case of big data and, as such, consists of heterogeneous streams that must be combined and transformed to yield consistent, comprehensive, current and correct information for business reporting and analysis. Data integration is complex for IoT data. There are many types of devices, most of which are not designed for compatibility with other systems. Data integration and the analytics that rely on it are two of the biggest challenges to IoT development.

Big data is sometimes characterized by the 3Vs model: Volume, variety and velocity. Volume refers to the amount of data, variety refers to the number of different types of data and devices, and velocity refers to the speed of data processing. The challenges of big data analytics – and IoT analytics – result from the simultaneous expansion of all three properties, rather than just the volume alone.

4.1.6 Apache Hadoop

Q7. What is MapReduce? Explain map reducer programming Model ?

Ans :

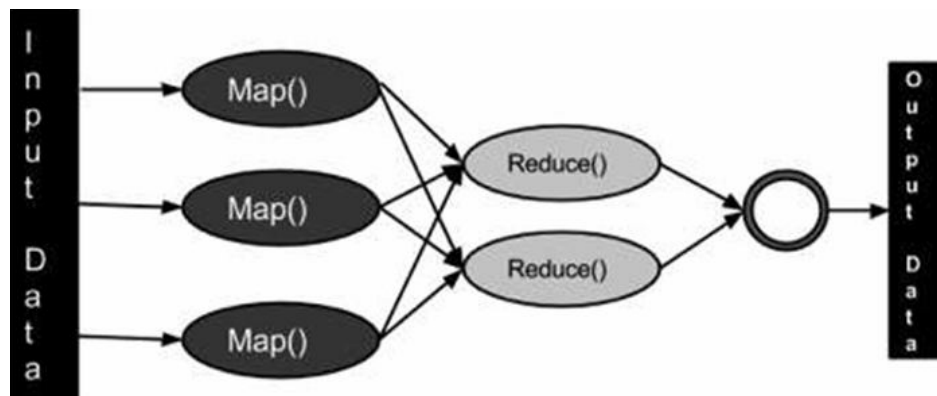
Map Reduce Programming Model

MapReduce is a processing technique and a program model for distributed computing based on java. The MapReduce algorithm contains two important tasks, namely Map and Reduce. Map takes a set of data and converts it into another set of data, where individual elements are broken down into tuples (key/value pairs). Secondly, reduce task, which takes the output from a map as an input and combines those data tuples into a smaller set of tuples. As the sequence of the name MapReduce implies, the reduce task is always performed after the map job.

The major advantage of MapReduce is that it is easy to scale data processing over multiple computing nodes. Under the MapReduce model, the data processing primitives are called mappers and reducers. Decomposing a data processing application into mappers and reducers is sometimes nontrivial. But, once we write an application in the MapReduce form, scaling the application to run over hundreds, thousands, or even tens of thousands of machines in a cluster is merely a configuration change. This simple scalability is what has attracted many programmers to use the MapReduce model.

- Generally MapReduce paradigm is based on sending the computer to where the data resides!
- MapReduce program executes in three stages, namely map stage, shuffle stage, and reduce stage.

- o **Map stage** : The map or mapper's job is to process the input data. Generally the input data is in the form of file or directory and is stored in the Hadoop file system (HDFS). The input file is passed to the mapper function line by line. The mapper processes the data and creates several small chunks of data.
 - o **Reduce stage** : This stage is the combination of the Shuffle stage and the Reduce stage. The Reducer's job is to process the data that comes from the mapper. After processing, it produces a new set of output, which will be stored in the HDFS.
- During a MapReduce job, Hadoop sends the Map and Reduce tasks to the appropriate servers in the cluster.
 - The framework manages all the details of data-passing such as issuing tasks, verifying task completion, and copying data around the cluster between the nodes.
 - Most of the computing takes place on nodes with data on local disks that reduces the network traffic.
 - After completion of the given tasks, the cluster collects and reduces the data to form an appropriate result, and sends it back to the Hadoop server.



Q8. Explain Map Reduce Job Execution with an example.

Ans :

Map reduce job execution

How Hadoop executes MapReduce (MapReduce v1) Jobs

To begin, a user runs a MapReduce program on the client node which instantiates a Job client object.

Next, the Job client submits the job to the JobTracker.

Then the job tracker creates a set of map and reduce tasks which get sent to the appropriate task trackers.

The task tracker launches a child process which in turns runs the map or reduce task.

Finally the task continuously updates the task tracker with status and counters and writes its output to its context.

A Word Count Example of MapReduce

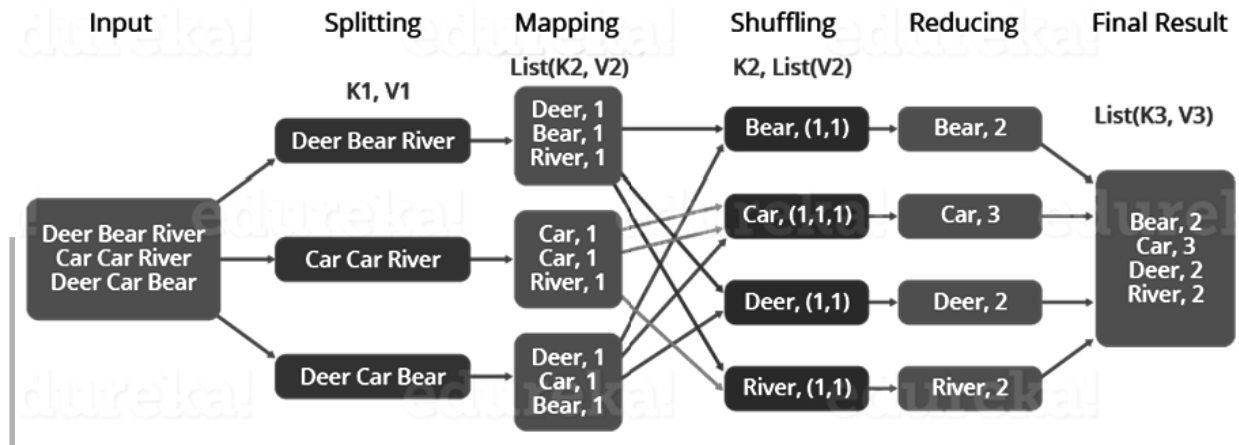
Let us understand, how a MapReduce works by taking an example where I have a text file called example.txt whose contents are as follows:

Dear, Bear, River, Car, Car, River, Deer, Car and Bear

Now, suppose, we have to perform a word count on the sample.txt using MapReduce. So, we will be finding the unique words and the number of occurrences of those unique words.

The Overall MapReduce Word Count Process

edureka!



- First, we divide the input in three splits as shown in the figure. This will distribute the work among all the map nodes.
- Then, we tokenize the words in each of the mapper and give a hardcoded value (1) to each of the tokens or words. The rationale behind giving a hardcoded value equal to 1 is that every word, in itself, will occur once.
- Now, a list of key-value pair will be created where the key is nothing but the individual words and value is one. So, for the first line (Dear Bear River) we have 3 key-value pairs – Dear, 1; Bear, 1; River, 1. The mapping process remains the same on all the nodes.
- After mapper phase, a partition process takes place where sorting and shuffling happens so that all the tuples with the same key are sent to the corresponding reducer.
- So, after the sorting and shuffling phase, each reducer will have a unique key and a list of values corresponding to that very key. For example, Bear, [1,1]; Car, [1,1,1].., etc.
- Now, each Reducer counts the values which are present in that list of values. As shown in the figure, reducer gets a list of values which is [1,1] for the key Bear. Then, it counts the number of ones in the very list and gives the final output as – Bear, 2.

Finally, all the output key/value pairs are then collected and written in the output file.

Q9. Explain Map reduce job execution work flow.

Ans :

Step by step MapReduce Job Flow

The data processed by MapReduce should be stored in **HDFS**, which divides the data into blocks and store distributed, Below are the steps for MapReduce data flow:

- **Step 1:** One block is processed by one mapper at a time. In the mapper, a developer can specify his own business logic as per the requirements. In this manner, Map runs on all the nodes of the cluster and process the data blocks in parallel.
- **Step 2:** Output of Mapper also known as intermediate output is written to the local disk. An output of mapper is not stored on HDFS as this is temporary data and **writing on HDFS** will create unnecessary many copies.
- **Step 3:** Output of mapper is shuffled to reducer node (which is a normal slave node but reduce phase will run here hence called as reducer node). The shuffling/copying is a physical movement of data which is done over the network.
- **Step 4:** Once all the mappers are finished and their output is shuffled on reducer nodes then this intermediate output is merged & sorted. Which is then provided as input to reduce phase.
- **Step 5:** Reduce is the second phase of processing where the user can specify his own custom business logic as per the requirements. An input to a reducer is provided from all the mappers. An output of reducer is the final output, which is written on HDFS.

Q10. What is a Hadoop Cluster? Explain.

Ans :

“A hadoop cluster is a collection of independent components connected through a dedicated network to work as a single centralized data processing resource.”

“A hadoop cluster can be referred to as a computational computer cluster for storing and analysing big data (structured, semi-structured and unstructured) in a distributed environment.”

“A computational computer cluster that distributes data analysis workload across various cluster nodes that work collectively to process the data in parallel.”

Hadoop clusters are also known as “Shared Nothing” systems because nothing is shared between the nodes in a hadoop cluster except for the network which connects them. The shared nothing paradigm of a hadoop cluster reduces the processing latency so when there is a need to process queries on huge amounts of data the cluster-wide latency is completely minimized.

Hadoop Cluster Architecture

A hadoop cluster architecture consists of a data centre, rack and the node that actually executes the jobs. Data centre consists of the racks and racks consists of nodes. A medium to large cluster consists of a two or three level hadoop cluster architecture that is built with rack mounted servers. Every rack of servers is interconnected through 1 gigabyte of Ethernet (1 GigE). Each rack level switch in a hadoop cluster is connected to a cluster level switch which are in turn connected to other cluster level switches or they uplink to other switching infrastructure.

Components of a Hadoop Cluster

Hadoop cluster consists of three components -

- **Master Node** – Master node in a hadoop cluster is responsible for storing data in HDFS and executing parallel computation the stored data using MapReduce. Master Node has 3 nodes – NameNode, Secondary NameNode and JobTracker. JobTracker monitors the parallel processing of data using MapReduce while the NameNode handles the data storage function with HDFS.

NameNode keeps a track of all the information on files (i.e. the metadata on files) such as the access time of the file, which user is accessing a file on current time and which file is saved in which hadoop cluster. The secondary NameNode keeps a backup of the NameNode data.

- **Slave/Worker Node-** This component in a hadoop cluster is responsible for storing the data and performing computations. Every slave/worker node runs both a TaskTracker and a DataNode service to communicate with the Master node in the cluster. The DataNode service is secondary to the NameNode and the TaskTracker service is secondary to the JobTracker.
- **Client Nodes –** Client node has hadoop installed with all the required cluster configuration settings and is responsible for loading all the data into the hadoop cluster. Client node submits mapreduce jobs describing on how data needs to be processed and then the output is retrieved by the client node once the job processing is completed.

Building a Hadoop Cluster

➤ Choosing the Right Hardware for a Hadoop Cluster

Many organizations are in a predicament when setting up hadoop infrastructure as they are not aware on what kind of machines they need to purchase for setting up an optimized hadoop environment and what is the ideal configuration they must use.

The number of machines or the hardware specification of machines depends on factors like –

- I. Volume of the Data
- II. The type of workload that needs to be processed (CPU driven or Use-Case/IO Bound)
- III. Data storage methodology (Data container , data compression technique used , if any)
- IV. Data retention policy (How long can you afford to keep the data before flushing it out)

➤ Sizing a Hadoop Cluster

The data volume that the hadoop users will process on the hadoop cluster should be a key consideration when sizing the hadoop cluster. Knowing the data volume to be processed helps decide as to how many nodes or machines would be required to process the data efficiently and how much memory capacity will be required for each machine

➤ Configuring the Hadoop Cluster

To obtain maximum performance from a Hadoop cluster, it needs to be configured correctly. However, finding the ideal configuration for a hadoop cluster is not easy.

4.1.7 YARN

Q11. What is YARN? Write about it?

Ans :

Apache Yarn – “Yet Another Resource Negotiator” is the resource management layer of **Hadoop**. The Yarn was introduced in Hadoop 2.x. Yarn allows different data processing engines like graph processing, interactive processing, stream processing as well as batch processing to run and process data stored in HDFS (Hadoop Distributed File System). Apart from resource management, Yarn is also used for job Scheduling. Yarn extends the power of Hadoop to other evolving technologies, so they can take the advantages of HDFS and economic cluster.

Apache yarn is also considered as the data operating system for Hadoop 2.x. The yarn based architecture of Hadoop 2.x provides a general purpose data processing platform which is not just limited to the MapReduce. It enables Hadoop to process other purpose-built data processing system other than MapReduce. It allows running several different frameworks on the same hardware where Hadoop is deployed.

Components of YARN

- o **Client:** For submitting MapReduce jobs.
- o **Resource Manager:** To manage the use of resources across the cluster
- o **Node Manager:** For launching and monitoring the computer containers on machines in the cluster.
- o **Map Reduce Application Master:** Checks tasks running the MapReduce job. The application master and the MapReduce tasks run in containers that are scheduled by the resource manager, and managed by the node managers.

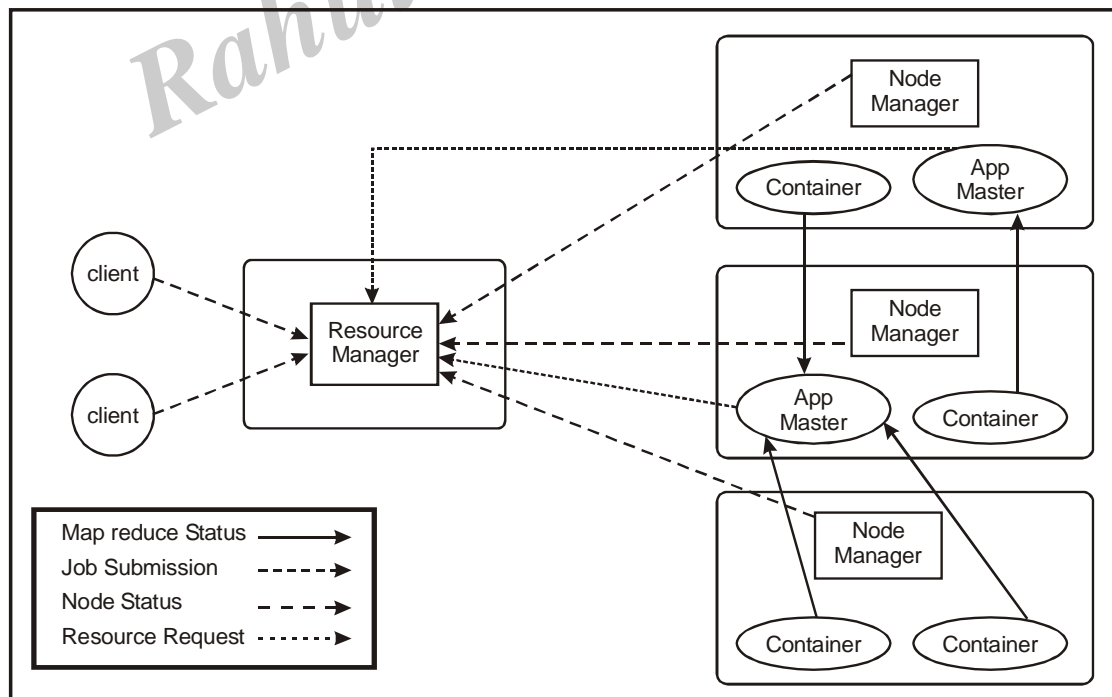
Jobtracker & Tasktracker were used in previous version of Hadoop, which were responsible for handling resources and checking progress management. However, Hadoop 2.0 has Resource manager and NodeManager to overcome the shortfall of Jobtracker&Tasktracker.

Q12. Describe the components of Hadoop Yarn Architecture.

Ans :

Yarn Architecture

Apache Yarn Framework consists of a master daemon known as "Resource Manager", slave daemon called node manager (one per slave node) and Application Master (one per application).



i. Resource Manager (RM)

It is the master daemon of Yarn. It manages the global assignments of resources (CPU and memory) among all the applications. It arbitrates system resources between competing applications. follow this Hadoop yarn Resource Manager guide to learn Yarn Resource manager in great detail.

Resource Manager has two Main components.

- ▶ Scheduler
- ▶ Application manager

a. Scheduler

The scheduler is responsible for allocating the resources to the running application. The scheduler is pure scheduler it means that it performs no monitoring no tracking for the application and even doesn't guarantees about restarting failed tasks either due to application failure or hardware failures.

b. Application Manager

It manages running Application Masters in the cluster, i.e., it is responsible for starting application masters and for monitoring and restarting them on different nodes in case of failures.

ii. Node Manager (NM)

It is the slave daemon of Yarn. NM is responsible for containers monitoring their resource usage and reporting the same to the Resource Manager. Manage the user process on that machine. Yarn NodeManager also tracks the health of the node on which it is running.

The design also allows plugging long-running auxiliary services to the NM; these are application-specific services, specified as part of the configurations and loaded by the NM during startup. For MapReduce applications on YARN, a shuffle is a typical auxiliary service loaded by the NMs. follow this Hadoop Yarn Node Manager guide to learn Node Manager in detail.

iii. Application Master (AM)

One application master runs per application. It negotiates resources from the resource manager and works with the node manager. It Manages the application life cycle.

The AM acquires containers from the RM's Scheduler before contacting the corresponding NMs to start the application's individual tasks.

Container

It is started by Node Manager. It consists of resources like memory, cpu core etc. For running a map or reduce task, Application Master asks Resource Manager for resources using which a container can be run.

4.1.8 OOZIE

Q13. What is Apache Oozie ? Explain the features and architecture.

Ans :

Apache Oozie is a scheduler system to run and **manage Hadoop jobs** in a distributed environment. It allows to combine multiple complex jobs to be run in a sequential order to achieve a bigger task. Within a sequence of task, two or more jobs can also be programmed to run parallel to each other.

One of the main advantages of Oozie is that it is tightly integrated with Hadoop stack supporting various Hadoop jobs like **Hive, Pig, Sqoop** as well as system-specific jobs like **Java and Shell**.

Oozie is an **Open Source Java Web-Application** available under Apache license 2.0. It is responsible for triggering the workflow actions, which in turn uses the Hadoop execution engine to actually execute the task. Hence, Oozie is able to leverage the existing Hadoop machinery for load balancing, fail-over, etc.

Oozie detects completion of tasks through callback and polling. When Oozie starts a task, it provides a unique **callback HTTP URL** to the task, and notifies that URL when it is complete. If the task fails to invoke the callback URL, Oozie can poll the task for completion.

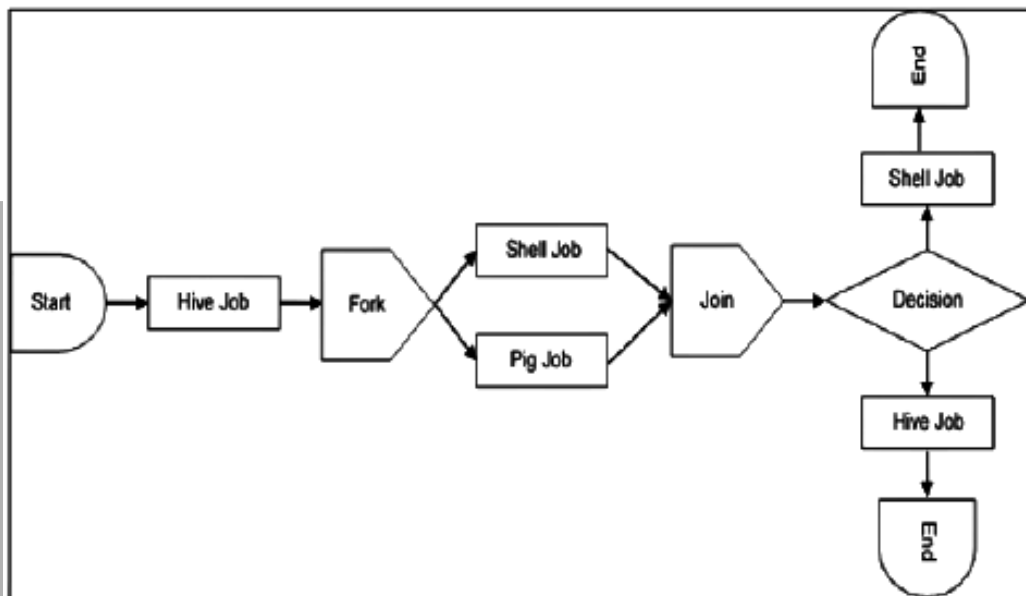
Following three types of jobs are common in Oozie :

- ▶ **Oozie Workflow Jobs** - These are represented as Directed Acyclic Graphs (DAGs) to specify a sequence of actions to be executed.

- ▶ **Oozie Coordinator Jobs** - These consist of workflow jobs triggered by time and data availability.
- ▶ **Oozie Bundle** - These can be referred to as a package of multiple coordinator and workflow jobs.

We will look into each of these in detail in the following chapters.

A sample workflow with Controls (Start, Decision, Fork, Join and End) and Actions (Hive, Shell, Pig) will look like the following diagram :



Workflow will always start with a Start tag and end with an End tag.

Features of Oozie

- ▶ Oozie has client API and command line interface which can be used to launch, control and monitor job from Java application.
- ▶ Using its Web Service APIs one can control jobs from anywhere.
- ▶ Oozie has provision to execute jobs which are scheduled to run periodically.
- ▶ Oozie has provision to send email notifications upon completion of jobs.

Oozie is a workflow/coordination system that you can use to manage Apache Hadoop jobs. It is one of the main components of Oozie is the Oozie server — a web application that runs in a Java servlet container (the standard Oozie distribution is using Tomcat). This server supports reading and executing Workflows, Coordinators, Bundles, and SLA definitions. It implements a set of remote Web Services APIs that can be invoked from Oozie client components and third-party applications.

Add a note hereThe execution of the server leverages a customizable database.

This database contains Workflow, Coordinator, Bundle, and SLA definitions, as well as execution states and process variables. The list of currently supported databases includes MySQL, Oracle, and Apache Derby.

The Oozie shared library component is located in the Oozie HOME directory and contains code used by the Oozie execution.

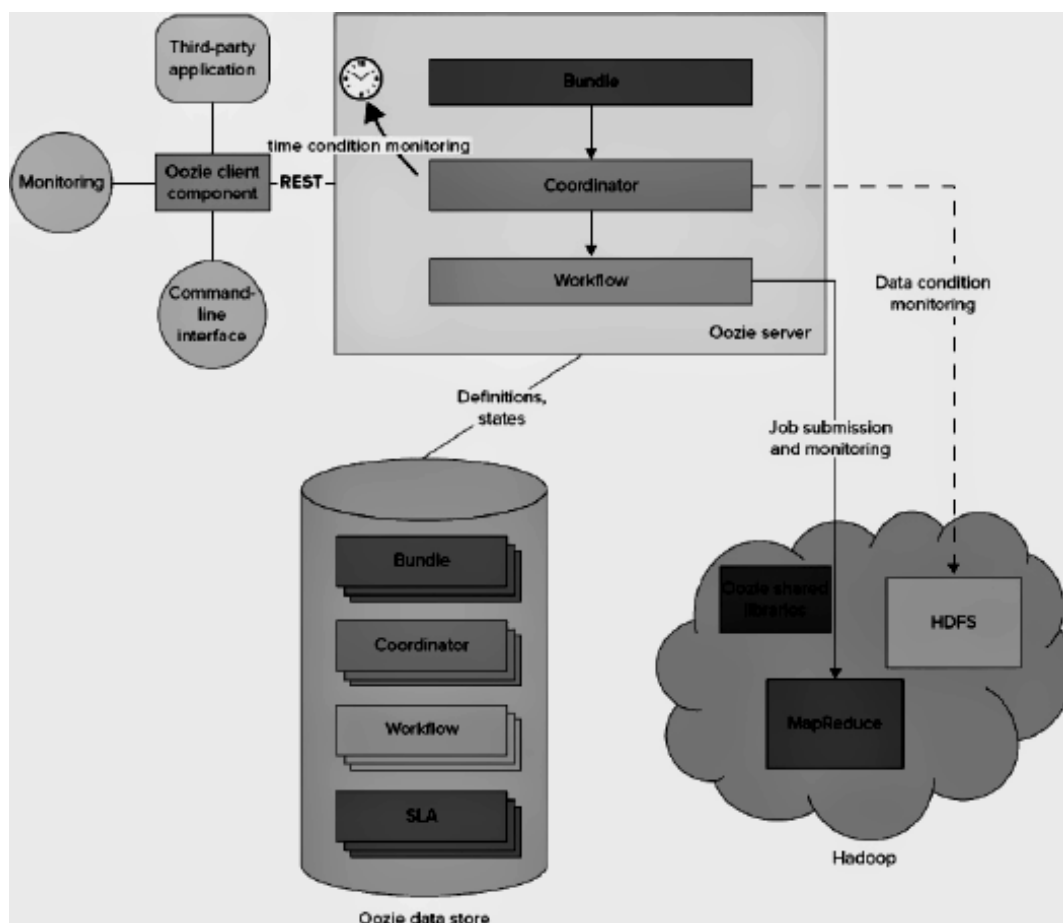
Oozie provides a command-line interface (CLI) that is based on a client component, which is a thin Java wrapper around Oozie Web Services APIs. These APIs can also be used from third-party applications that have sufficient permissions.

A single Oozie server implements all four functional Oozie components:

Oozie Workflow

- ▶ Oozie Coordinator
- ▶ Oozie Bundle
- ▶ Oozie SLA
- ▶ Oozie server is described in this chapter, starting with what Oozie Workflow is and how you can use it.

Oozie Architecture



4.1.9 Spark

Q14. What is Apache Spark? Explain about it.

Ans :

Apache Spark is a lightning-fast cluster computing technology, designed for fast computation. It is based on Hadoop MapReduce and it extends the MapReduce model to efficiently use it for more types of computations, which includes interactive queries and stream processing. The main feature of Spark is its **in-memory cluster computing** that increases the processing speed of an application.

Spark is designed to cover a wide range of workloads such as batch applications, iterative algorithms, interactive queries and streaming. Apart from supporting all these workload in a respective system, it reduces the management burden of maintaining separate tools.

Evolution of Apache Spark

Spark is one of Hadoop's sub project developed in 2009 in UC Berkeley's AMPLab by Matei Zaharia. It was Open Sourced in 2010 under a BSD license. It was donated to Apache software foundation in 2013, and now Apache Spark has become a top level Apache project from Feb-2014.

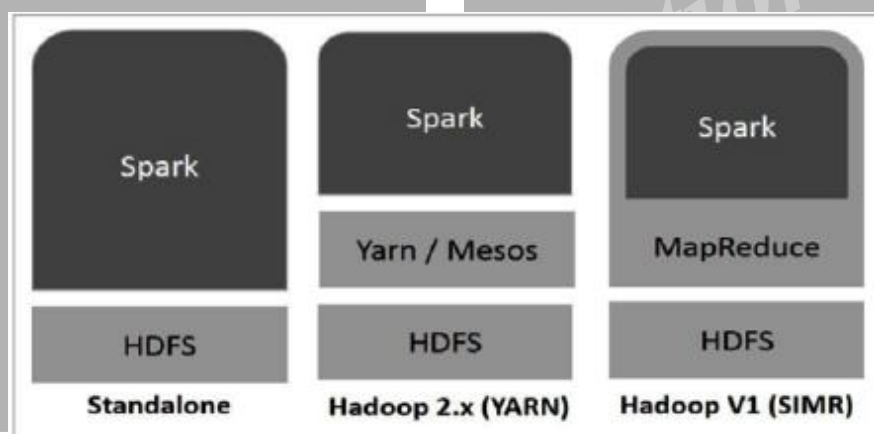
Features of Apache Spark

Apache Spark has following features.

- ▶ **Speed**- Spark helps to run an application in Hadoop cluster, up to 100 times faster in memory, and 10 times faster when running on disk. This is possible by reducing number of read/write operations to disk. It stores the intermediate processing data in memory.
- ▶ **Supports multiple languages** - Spark provides built-in APIs in Java, Scala, or Python. Therefore, you can write applications in different languages. Spark comes up with 80 high-level operators for interactive querying.
- ▶ **Advanced Analytics** - Spark not only supports 'Map' and 'reduce'. It also supports SQL queries, Streaming data, Machine learning (ML), and Graph algorithms.

Spark Built on Hadoop

The following diagram shows three ways of how Spark can be built with Hadoop components.



There are three ways of Spark deployment as explained below.

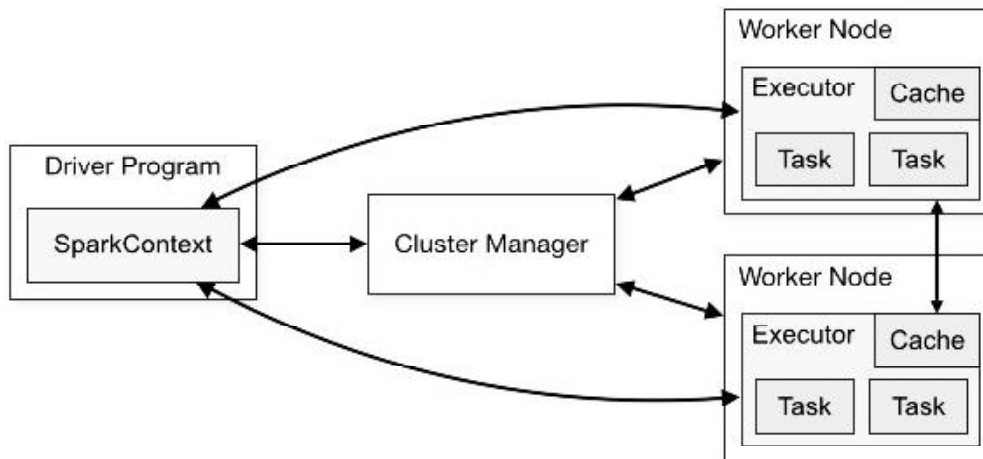
- ▶ **Standalone** - Spark Standalone deployment means Spark occupies the place on top of HDFS(Hadoop Distributed File System) and space is allocated for HDFS, explicitly. Here, Spark and MapReduce will run side by side to cover all spark jobs on cluster.
- ▶ **Hadoop Yarn** - Hadoop Yarn deployment means, simply, spark runs on Yarn without any pre-installation or root access required. It helps to integrate Spark into Hadoop ecosystem or Hadoop stack. It allows other components to run on top of stack.
- ▶ **Spark in MapReduce (SIMR)** - Spark in MapReduce is used to launch spark job in addition to standalone deployment. With SIMR, user can start Spark and uses its shell without any administrative access.

Apache Spark Components and Architecture

SparkContext is an independent process through which spark application runs over a cluster. It gives the handle to the distributed mechanism/cluster so that you may use the resources of the distributed

machines in your job. Your application program which will use SparkContext object would be known as driver program. Specifically, to run on a cluster, the SparkContext connects to several types of cluster managers (like Spark's own standalone cluster manager, apache Mesos or Hadoop's YARN), which allocate resources across applications. Once connected, Spark takes over executors on distributed nodes in the cluster, which are processes in the distributed nodes that run computations and store data for your application. Next, it sends your application code to the executors through SparkContext. Finally tasks are sent to the executors to run and complete it.

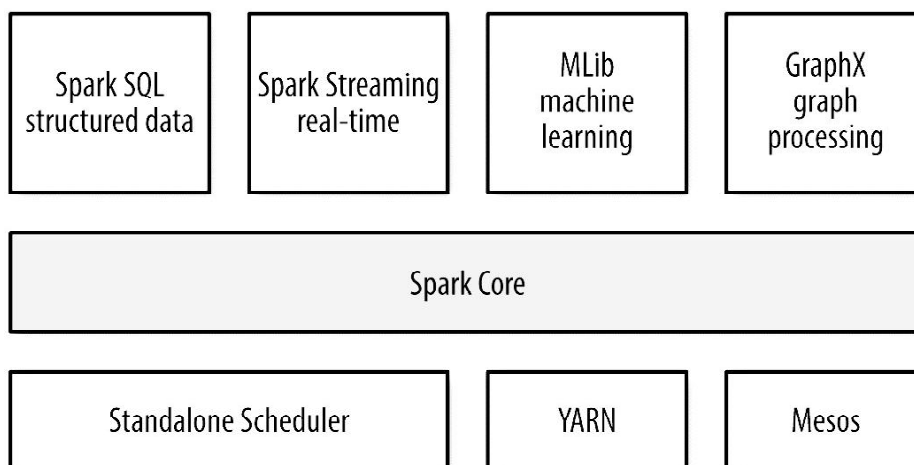
Cluster overview



Following are most important takeaways of the architecture :

- ▶ Each application gets its own executor processes, which remains in memory up to the duration of the complete application and run tasks in multiple threads. This means each application is independent from the other, on both the scheduling side since each driver schedules its own tasks and executor side as tasks from different applications run in different JVMs.
- ▶ Spark is independent of cluster managers that implies, it can be coupled with any cluster manager and then leverage that cluster.
- ▶ Because the driver schedules tasks on the cluster, it should be run as close to the worker nodes as possible.

Spark Eco-sytem components



Spark Core

Spark Core is the base of an overall spark project. It is responsible for distributed task dispatching, parallelism, scheduling, and basic I/O functionalities. All the basic functionality of spark core are exposed through an API (for Java, Python, Scala, and R) centered on the RDD abstraction. A 'driver' program starts parallel operations such as map, filter or reduce on any RDD by passing a function to SparkCore, which further schedules the function's execution in parallel on the cluster.

Other than Spark Core API, there are additional useful and powerful libraries that are part of the Spark ecosystem and adds powerful capabilities in Big Data analytics and Machine Learning areas. These libraries include:

► Spark Streaming

Spark Streaming is a useful addition to the core Spark API. It enables high-throughput, fault-tolerant stream processing of live data streams. It is used for processing real-time streaming data. This is based on micro batch style of computing and processing. The fundamental stream unit is DStream. DStream is basically a series of RDDs, to process the real-time data.



Spark SQL, DataFrames and Datasets :

► SQL

Spark SQL exposes spark APIs to run SQL query like computation on large data. A spark user can perform ad-hoc query and perform near real time ETL on a different types of data like (like JSON, Parquet, Database).

► DataFrames

A DataFrame can be considered as a distributed set of data which has been organized into many named columns. It can be compared with a relational table, CSV file or a data frame in R or Python. The DataFrame functionality is made available as API in Scala, Java, Python, and R.

► Dataset

A Dataset is a new addition in the list of spark libraries. It is an experimental interface added in Spark 1.6 that tries to provide the benefits of RDDs with the benefits of Spark SQL's optimized execution engine.

► Spark MLlib And ML

MLlib is collective bunch few handy and useful machine learning algorithms and data cleaning and processing approaches which includes classification, clustering, regression, feature extraction, dimensionality reduction, etc. as well as underlying optimization primitives like SGD and BFGS.

► Spark GraphX

GraphX is the Spark API for graphs and graph-parallel computation. GraphX enhances the Spark RDD by introducing the Resilient Distributed Property Graph.

A RDD property graph is a directed multi-graph with properties attached with each of its vertex and edge. GraphX has a set of basic operators (like subgraph, joinVertices, aggregateMessages, etc.). Along with operators it has an optimized variant of the Pregel API. GraphX is still under development and many developers are working towards simplification of graph related tasks.

4.1.10 Storm

Q15. What is Apache Storm? Explain.

Ans :

Apache Storm is a distributed real-time big data-processing system. Storm is designed to process vast amount of data in a fault-tolerant and horizontal scalable method. It is a streaming data framework that has the capability of highest ingestion rates. Though Storm is stateless, it manages distributed environment and cluster state via Apache ZooKeeper. It is simple and you can execute all kinds of manipulations on real-time data in parallel.

Apache Storm is continuing to be a leader in real-time data analytics. Storm is easy to setup, operate and it guarantees that every message will be processed through the topology at least once.

Apache Storm vs Hadoop

Basically Hadoop and Storm frameworks are used for analyzing big data. Both of them complement each other and differ in some aspects. Apache Storm does all the operations except persistency, while Hadoop is good at everything but lags in real-time computation. The following table compares the attributes of Storm and Hadoop.

Storm	Hadoop
Real-time stream processing	Batch processing
Stateless	Stateful
Master/Slave architecture with ZooKeeper based coordination. The master node is called as nimbus and slaves are supervisors .	Master-slave architecture with/without ZooKeeper based coordination. Master node is job tracker and slave node is task tracker .
A Storm streaming process can access tens of thousands messages per second on cluster.	Hadoop Distributed File System (HDFS) uses MapReduce framework to process vast amount of data that takes minutes or hours.
Storm topology runs until shutdown by the user or an unexpected unrecoverable failure.	MapReduce jobs are executed in a sequential order and completed eventually.
Both are distributed and fault-tolerant	
If nimbus / supervisor dies, restarting makes it continue from where it stopped, hence nothing gets affected.	If the JobTracker dies, all the running jobs are lost.

Use-Cases of Apache Storm

Apache Storm is very famous for real-time big data stream processing. For this reason, most of the companies are using Storm as an integral part of their system. Some notable examples are as follows:

- ▶ **Twitter** - Twitter is using Apache Storm for its range of “Publisher Analytics products”. “Publisher Analytics Products” process each and every tweets and clicks in the Twitter Platform. Apache Storm is deeply integrated with Twitter infrastructure.
- ▶ **NaviSite** - NaviSite is using Storm for Event log monitoring/auditing system. Every logs generated in the system will go through the Storm. Storm will check the message against the configured set of regular expression and if there is a match, then that particular message will be saved to the database.
- ▶ **Wego** - Wego is a travel metasearch engine located in Singapore. Travel related data comes from many sources all over the world with different timing. Storm helps Wego to search real-time data, resolves concurrency issues and find the best match for the end-user.

Apache Storm Benefits

Here is a list of the benefits that Apache Storm offers :

- ▶ Storm is open source, robust, and user friendly. It could be utilized in small companies as well as large corporations.
- ▶ Storm is fault tolerant, flexible, reliable, and supports any programming language.
- ▶ Allows real-time stream processing.

- ▶ Storm is unbelievably fast because it has enormous power of processing the data.
- ▶ Storm can keep up the performance even under increasing load by adding resources linearly. It is highly scalable.
- ▶ Storm performs data refresh and end-to-end delivery response in seconds or minutes depends upon the problem. It has very low latency.
- ▶ Storm has operational intelligence.
- ▶ Storm provides guaranteed data processing even if any of the connected nodes in the cluster die or messages are lost.

Apache Storm: Architecture

Components of a Storm Cluster

An Apache Storm cluster is superficially similar to a Hadoop cluster. Whereas on Hadoop you run MapReduce jobs, on Storm, you run topologies. Jobs and topologies themselves are very different — one key difference being that a MapReduce job eventually finishes, whereas a topology processes messages forever (or until you kill it).

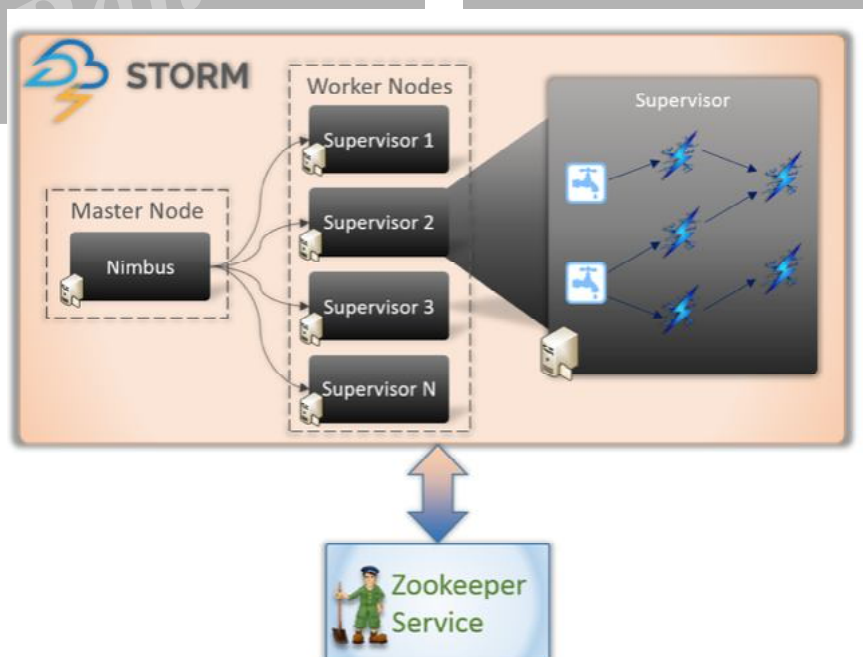
There are two kinds of nodes in a Storm cluster: master node and worker nodes.

1. **Master Node (Nimbus).** The master node runs a daemon called Nimbus that is similar to Hadoop's JobTracker. Nimbus is responsible for distributing code around the cluster, assigning tasks to machines, and monitoring for failures.

Nimbus is an Apache Thrift service enabling you to submit code in any programming language. This way, you can always utilize the language that you are proficient in without needing to learn a new language to utilize Apache Storm.

The Nimbus service relies on Apache ZooKeeper to monitor the message processing tasks as all the worker nodes update their tasks status in the Apache ZooKeeper service.

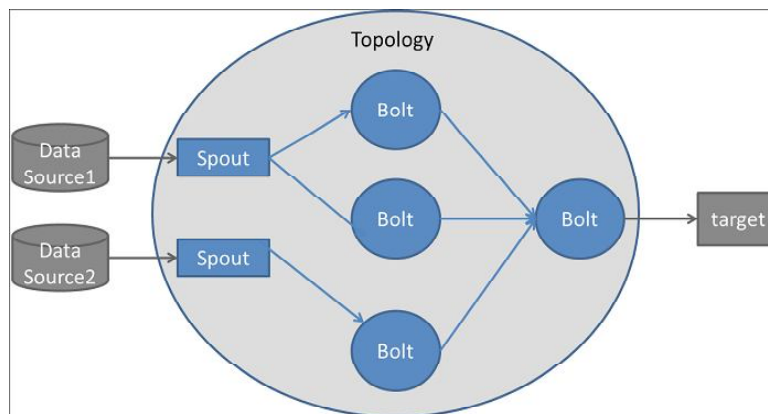
2. **Worker Nodes (Supervisor).** Each worker node runs a daemon called the Supervisor. The supervisor listens for work assigned to its machine and starts and stops worker processes as necessary based on what Nimbus has assigned to it. Each worker process executes a subset of a topology; a running topology consists of many worker processes spread across many machines.



All coordination between Nimbus and the Supervisors is done through a ZooKeeper cluster. Additionally, the Nimbus daemon and Supervisor daemons are fail-fast and stateless. Even though stateless nature has its own disadvantages, it actually helps Storm process real-time data in the best possible and quickest way.

Storm is not entirely stateless, though. It stores its state in Apache ZooKeeper. Since the state is available in Apache ZooKeeper, a failed Nimbus can be restarted and made to work from where it left. Service monitoring tools can monitor Nimbus and restart it if there is any failure.

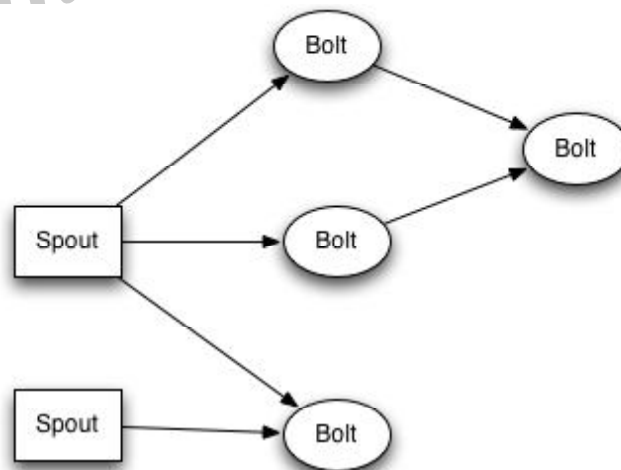
Apache Storm also has an advanced topology called Trident Topology with state maintenance, and it also provides a high-level API like Pig.



Topologies

To do real-time computation on Storm, you create what are called topologies. A topology is a graph of computation and is implemented as DAG (directed acyclic graph) data structure.

Each node in a topology contains processing logic (bolts) and links between nodes indicate how data should be passed around between nodes (streams).



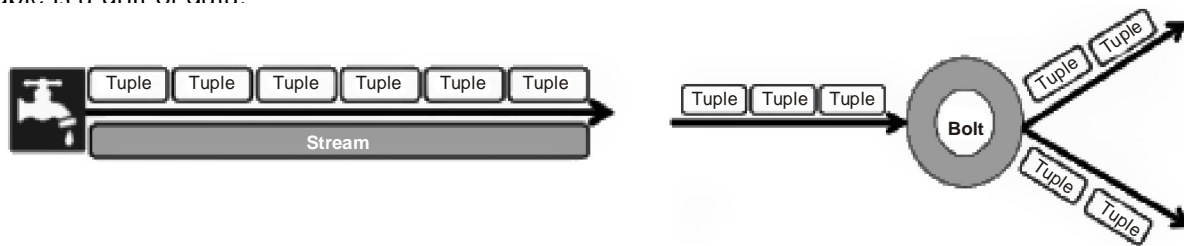
When a topology is submitted to a Storm cluster, the Nimbus service on master node consults the supervisor services on different worker nodes and submits the topology. Each supervisor creates one or more worker processes, each having its own separate JVM. Each process runs within itself threads that we call Executors.

The thread/executor processes the actual computational tasks: Spout or Bolt.

Running a topology is straightforward. First, you package all your code and dependencies into a single JAR. Then, you run a command like the following:

```
storm jar all-my-code.jar org.apache.storm.MyTopology arg1 arg2
```

Streams represent the unbounded sequences of tuples (collection of key-value pairs) where a tuple is a unit of data.



A stream of tuples flows from spout to bolt(s) or from bolt(s) to another bolt(s). There are various stream grouping techniques to let you define how the data should flow in topology like global grouping, etc.

Spouts

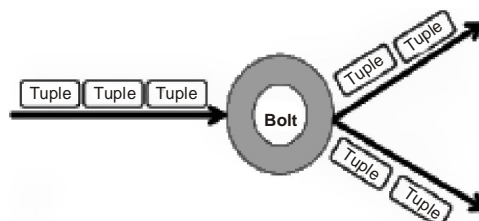
A spout is the entry point in a Storm topology. It represents the source of data in Storm. Generally, spouts will read tuples from an external source and emit them into the topology. You can write spouts to read data from data sources such as a database, distributed file systems, messaging frameworks, or a message queue as Kafka from where it gets continuous data, converts the actual data into a stream of tuples, and emits them to bolts for actual processing. Spouts run as tasks in worker processes by Executor threads.

Spouts can broadly be classified as follows :

- ▶ **Reliable:** These spouts have the ability to replay the tuples (a unit of data in the data stream). This helps applications achieve the “at least once message processing” semantic as, in case of failures, tuples can be replayed and processed again. Spouts for fetching data from messaging frameworks are generally reliable, as these frameworks provide a mechanism to replay the messages.
- ▶ **Unreliable:** These spouts don't have the ability to replay the tuples. Once a tuple is emitted, it cannot be replayed, regardless of whether it was processed successfully. This type of spout follows the “at most once message processing” semantic.

Bolts

All processing in topologies is done in bolts. Bolts can do anything from filtering and functions to aggregations, joins, talking to databases, and more. Bolts can do simple stream transformations. Doing complex stream transformations often requires multiple steps and thus multiple bolts. For example, transforming a stream of tweets into a stream of trending images requires at least two steps: a bolt to do a rolling count of retweets for each image and one or more bolts to stream out the top X images (you can do this particular stream transformation in a more scalable way with three bolts than with two).



Bolts can also emit more than one stream.

4.1.11 Health Monitoring Case Study

Q16. Write a case study on health monitoring system.

Ans :

There have been many efforts in the field of IoT based remote patient monitoring systems. Piccini et al. discuss wireless system based on Bluetooth for acquiring bio-medical signals such as Electrocardiography (ECG), Electromyography (EMG), Electroencephalography (EEG) and Electrooculography (EOG). The architecture consists of two operational units: one to acquire single lead ECG signal and the other a DSP system to clean the acquired signal from the first unit. The system reads signals including ECG, EMG, EEG and EOG, heart rate, breathing and blood pressure, processes it and sends it to a remote server or displays it over LCD screen. The system implements priority scheduling and data compression, which reduces the transmission delays of critical signals and saves bandwidth and power.

System Architectures for Health Monitoring

Here with, we discuss the implementation of two architectures for remote monitoring of bio-medical signals. Medical applications have certain nature and requirements that usually have life or death consequences when data is not successfully transferred (e.g. lost, corrupted, delayed, etc.) as opposed to most other applications where requirements and concerns are mostly financial. These requirements such as data rate and delay have been defined by the IEEE 1073 group. For example, in case of 3-lead ECG system, a patient node (i.e., a wireless electrode) generates 2.4 Kbits/s of data [6]. In our implementations, the sensors used to collect medical data include Blood Pressure, Heart Rate, Temperature, Respiration, Glucose, SpO2, and ECG. Data rate for bio-medical signal varies significantly. The data rates of various signals are presented in Table 1.

Bio-medical Signal	Latency	Data Rate
Blood pressure	< 3 s	80 - 800 bps
Pulse / Heart Rate	< 3 s	80 - 800 bps
Glucose	< 3 s	80 - 800 bps
Temperature	< 3 s	80 - 800 bps
Respiration	< 300 ms	50 - 120 bps
SpO2	< 300 ms	50 - 120 bps
ECG	< 300 ms	3-lead (2.4 kbps), 5-lead (10 kbps), 12-lead (72 kbps).

Table : Data rate of various bio-medical signals

The first architecture implements wireless sensor network based on low-power ZigBee, while the second architecture implements IP-based wireless sensor network using Wi-Fi.

A. ZigBee-Based Architecture

ZigBee is based on low-rate IEEE 802.15.4 standard, designed for supporting low-power, low-cost, and low-data rate applications. The ZigBee based architecture consists of several patient nodes and a sink node. The system is implemented with ZigduinoR2 hardware platform, which is an Arduino compatible microcontroller platform. ZigBee based architecture as shown in Figure 1 can be divided into four sections; sensor interface, WSN implementation, database application and webserver application.

Sensor interface: The sensor interface is implemented using an Arduino-compatible E-health shield on top of the Zigduino hardware. The E-health shield is basically a gateway between the medical sensors and Zigduino board. Data measured from various sensors are collected by the Zigduino board via the E-health shield.

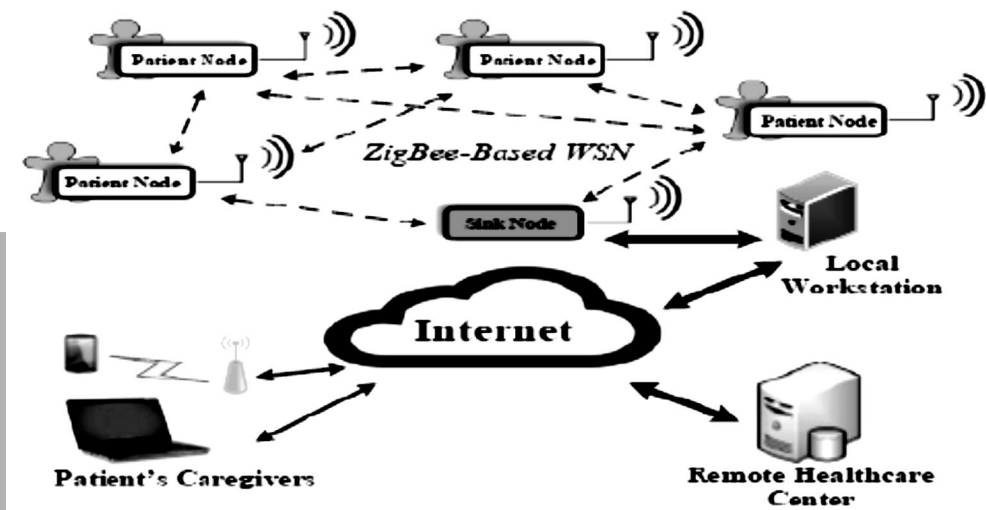


Fig. : ZigBee Based Health Monitoring System

WSN Implementation

The Zigduino's microcontroller contains an on-chip 2.4 GHz IEEE 802.15.4 radio. The implemented WSN consists of several patient (client) nodes and a sink (server) node. Patient nodes collect data from various sensors and send wirelessly over ZigBee to the sink (server) node. The code architecture of sink and patient nodes are shown in Table 2.

Server (sink) node architecture	
ZigBeeServer	Send and receive data over ZigBee
ServiceServer	Add and remove nodes in the network and assign ID to them
MACServer	Grants permission to the nodes to access media.
Client (Patient) node architecture	
ZigBeeServer	Send and receive data over ZigBee
MeasurementServer	Collect data and store them in FIFO
ServiceServer	Add and remove nodes in the network and assign ID to them
MACServer	Grants permission to the nodes to access media

Table : Code Architecture of sink and patient node

- ▶ **Database application :** The sink (server) node is connected to a local PC (Personal computer) where a Python code executes to collect data from the serial terminal and save it into a remote database.
- ▶ **Webserver Application:** Web-server application written with PHP accesses the database and updates the web page in real time. The data from the webpage can be accessed remotely by patient's caregivers through their laptops or smart phones using any browser.

4.2 AN IOT TOOL

4.2.1 CHEF

Q17. What is Chef ? Explain it.

Ans :

Chef is a configuration management tool that is written in Ruby and Erlang. It's capable of managing both your on premise and cloud servers with ease.

You can easily manage up to 10000 nodes using chef. Replicating the infrastructure components is easy once we have them automated via chef.

Chef has three core components which are mentioned below.

- ▶ **Chef Server :** This central server holds all configuration data that the nodes will use for configuration.
- ▶ **Workstation :** This machine holds all the configuration data that can later be pushed to the central chef server. Several chef command line utilities will be available in this system, which can be used to interact with nodes, update configurations etc. This is the place from which most of the work happens on a day to day basis.

- ▶ **Node :** This is nothing but a client server/ system that will be registered to the central chef server, from where it can pull configuration data that needs to be applied.

Let's understand each of the above mentioned components in a bit more detail.

Central Chef Server

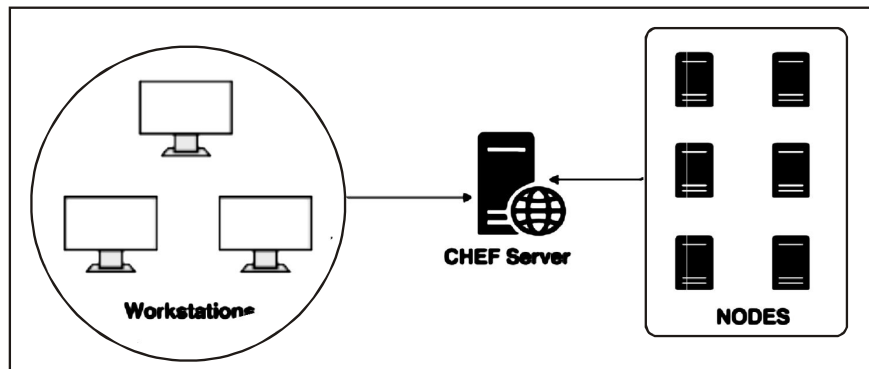
This is a centrally located server that holds all details related to chef infrastructure. These details include full metadata of all the clients that are automated via chef. All configurations applicable to different clients in the architecture.

Chef runs in a server client mode. Each node has a chef client software installed, which will pull down the configuration that are applicable to that node from the central chef server.

The central chef server has an optional web interface which provides several administrative capabilities to users managing chef. Nodes can be deleted and configurations applicable to a node can be modified using this central web interface.

There are three different types of chef server available.

- ▶ **Chef Solo :** Actually chef solo is not a chef server. In fact it removes the need of having a central chef server to test configurations on nodes.
- ▶ **Open Source Chef :** This is completely free and open source chef, which you can install anywhere.
- ▶ **Hosted Chef :** This is paid, where opsource will manage your central chef server, which you can access/configure using the web interface. This makes you free from the responsibility of managing a central chef server yourself.



Workstation

Consider workstation as a system that can be used to control central chef server. As depicted in the above diagram, there can be multiple workstations that can together manage a central chef server.

Workstations will do the below jobs.

- ▶ Writing cookbooks and recipes that will later be pushed to central chef server.

A cookbook is nothing but a unit that configures a particular thing on the node.

- ▶ Managing Nodes on the central chef server.

The workstation system will have the required command line utilities, to control and manage every aspect of the central chef server. Things like adding a new node to the central chef server, deleting a node from the central chef server, modifying node configurations etc can all be managed from the workstation itself.

Basically workstation will have two main components.

1. **Knife utility :** This command line tool can be used to communicate with the central chef server from workstation. Adding, removing, changing configurations, of nodes in central chef server will be carried out by using this knife utility.

Cookbooks can be uploaded to central chef server using knife utility, Roles and environments can be managed using knife utility. Basically every aspect of central chef server can be controlled from workstation using knife utility.

2. **A local Chef repository :** This is the place where every configuration components of central chef server is stored. This chef repository can be synchronized with the central chef server (again using the knife utility itself.)

Nodes

Nodes can be a cloud based virtual server or a physical server in your own data centre, that is managed using central chef server. The main component that needs to be present on the node is an agent that will establish communication with the central chef server.

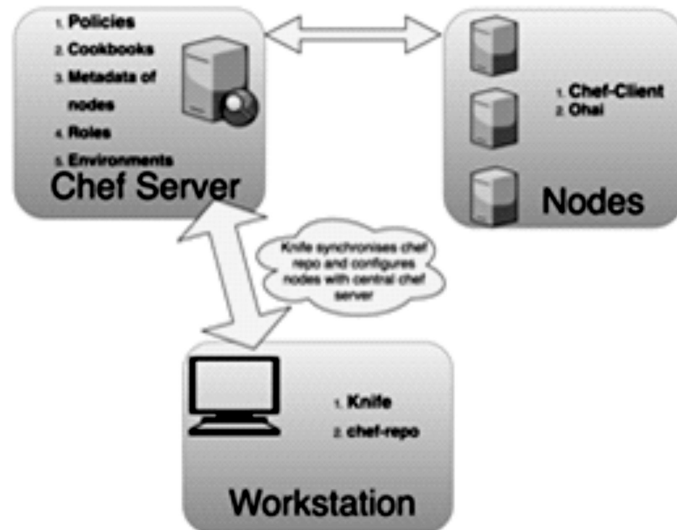
This agent is called as Chef client.

Chef client does the following...

- ▶ Its responsible for interacting with the central chef server.
- ▶ It manages the initial registration of the node to the central chef server.

- It pulls down cookbooks, and applies them on the node, to configure the node.
- Periodic polling of the central chef server to fetch new configuration items if any.

In order for the chef client to configure the node depending upon the cookbooks it pulls down from central chef server, chef client needs a lot of details about the node.



Q18. Explain how to setup chef tool.

Ans :

Installing Chef Server

Step 1: The very first step is to verify that correct hostname is configured on the server. And the hostname FQDN is actually resolvable. Its very much necessary and recommended to have proper DNS entries in place for your chef server. If you are not in a production environment, you can get it done using hosts file as well. Login to the server, and run the below commands to set correct hostname..

```
1 hostnamechef.example.com
echo"chef.example.com"> /etc/hostname
```

In the above example, am using example.com for demonstration purposes only. You will have to replace it with the domain name applicable to your environment. For this example, we will be adding an entry in /etc/hosts file. As mentioned earlier, for production environments, you need to have proper DNS entry for your chef server FQDN.

```
10.12.2.188 chef.example.com
```

Add the above entry inside /etc/hosts file. In the above example, i have used 10.12.2.188 because that's the ip address of my server. Replace it with the IP address of your server while making the entry inside /etc/hostfile.

If your entry is correct, you should be able to ping chef.example.com (replace it with your dns name).

Step 2 : The second step is to ensure time synchronization. So we need to configure NTP on the server. This can be done by installing ntp package in Ubuntu as shown below..

```
apt-get update
apt-get installntp
service ntp start
```

Step 3: The next step is to download Chef server package from official website and install it. For this, access the below URL for chef server download page.

Right Click on the 64 bit package and select "Copy Link Address" as shown below.

Chef Server

The Chef server makes it easy to automate your infrastructure, manage scale and complexity, and safeguard your systems.



Step 4: Paste the URL we copied in step 3 as a parameter to wget command. Wget will download this package to our Ubuntu 14.04 server. If you are new to wget, I would recommend reading the below article for its complete usage.

```
wget https://packages.chef.io/stable/ubuntu/14.04/chef-server-core_12.7.0-1_amd64.deb
```

Step 5: The next step is to install this downloaded package using the below command.

```
dpkg -i chef-server*
```

Once the above command succeeds, you will have chef server installed on the system. We still need to configure it as shown in the below pending steps.

Step 6: The next step is to ask chef server to configure itself with the default settings. This can be done by running the below command.

```
chef-server-ctl reconfigure
```

This command will take a while to complete. This is because it installs and configures all the previously mentioned components of chef server (ie: Nginx, RabbitMQ, Postgresql, Solr etc.). The successful completion of the above command should show you something like the below towards the end..

Chef Client finished, 39/411 resources updated in 31 seconds

Chef Server Reconfigured!

Step 7: Once the above command is completed successfully, we have chef server components installed and configured on the server. You can actually have an optional web interface for chef as well. This web interface is called as chef manage. It can be installed using the below command.

Please keep the fact in mind that chef manage web interface is only free for up to 25 nodes.

```
chef-server-ctl install chef-manage
```

```
chef-server-ctl reconfigure
```

```
chef-manage-ctl reconfigure
```

Once the above three commands succeed, you have the web interface called chef manage ready for you to use. You can access the web interface by either using the IP address of your server or the DNS name of the server (if you have DNS configured correctly in your environment.)

Before we can access the web interface, we need to create an admin user and credentials. This can be done by executing the below command.

```
chef-server-ctl user-create sarathSarathPillai sarath@slashroot.in password —filename sarath.pem
```

“sarath” in the above command is the username.

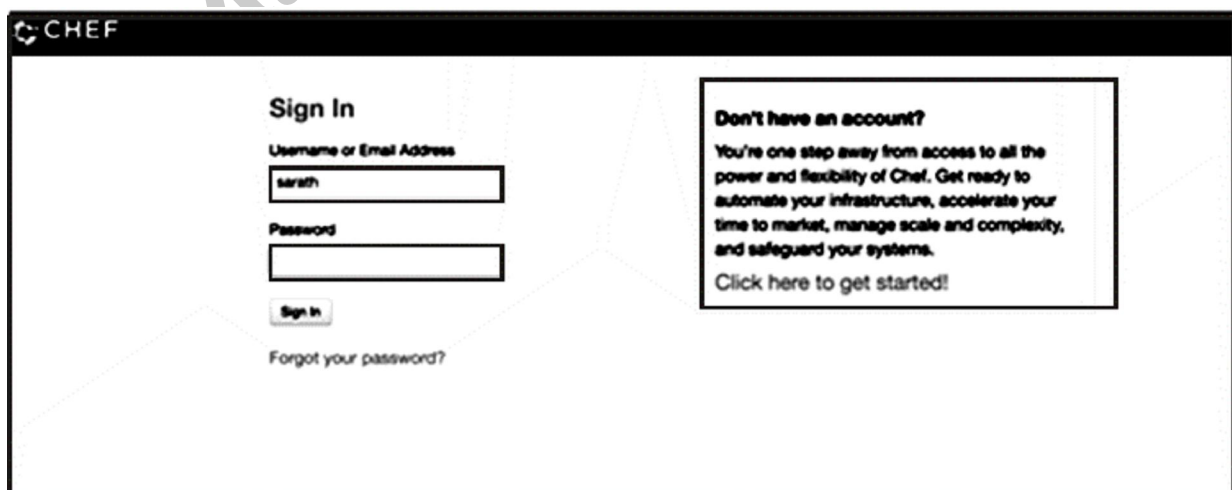
“SarathPillai” is the full name of the user.

“sarath@slashroot.in” is the email address.

“password” is the actual password for the user. Replace it with something complex

“sarath.pem” is the private key file for the user (this will be created in the current directory after the command completes.)

You can now access the web interface by using the IP address of the server in the web browser. The login page looks like the below.



You can enter the username that we created in the previous step here, along with the password we created. Once you enter the credentials, you will be greeted with the below message.



You can now create an organization by using the "Create New Organization" link shown above. Once you click on that link, you should then be able to create the organization as shown below.

The screenshot shows a "Create Organization" dialog box. It has a title bar with "Create Organization" and a close button (X). Inside, there are two text input fields. The first is labeled "Full Name (example: Chef, Inc.)" and contains the text "Slashroot, Inc.". The second is labeled "Short Name (example: chef)" and contains the text "slashroot". At the bottom right, there are two buttons: "Cancel" and "Create Organization".

4.2.2 Chef Case Studies

Q19. Create Hadoop Cluster using Chef.

Ans :

Hadoop Cluster Setup

Chef-bach can be used to create a hadoop test cluster using virtual machines on an hypervisor host with enough resources. The resulting cluster will be a 4 node cluster with one of the nodes acting as

the bootstrap node which will host a chef server. The other three nodes will be hadoop nodes. 2 out of 3 nodes will be master nodes and one node will be the worker node. The following are the steps to go about creating the test cluster. This has been tested on hypervisor hosts running Mac OS and Ubuntu.

- ▶ Install curl on the hypervisor host
- ▶ Install virtualbox on the hypervisor host (get the latest from VirtualBox.ORG)
 - o Likely, this will need make, gcc too
- ▶ Install vagrant on the hypervisor host (get the latest from VagrantUp.COM)
- ▶ Delete the default DHCP server inbuilt in virtualbox.

```
$ vboxmanage list dhcpservers
```

```
NetworkName: HostInterfaceNetworking-  
vboxnet0
```

```
IP: 192.168.56.100
```

```
NetworkMask: 255.255.255.0
```

```
lowerIPAddress: 192.168.56.101
```

```
upperIPAddress: 192.168.56.254
```

```
Enabled: Yes
```

```
$ vboxmanagedhcpserver remove --  
netname HostInterfaceNetworking-vboxnet0
```

Note: Run the above two commands with sudo too in case the current user does not have access to view/edit the dhcpservers

- ▶ Run `sudo kill -f VBox` on the hypervisor host
- ▶ Clone chef-bach repository onto the hypervisor host `git clone https://github.com/bloomberg/chef-bach.git`
- ▶ rename chef-bach to chef-bcpc directory on the hypervisor host
- ▶ cd to chef-bcpc directory on the hypervisor host
- ▶ Run the auto installation script under the test directory `./tests/automated_install.sh`
- ▶ This will download all the required software, creates the four node cluster and installs all

the HDP hadoop components. As you can imagine this takes some time. Depending on the size of the hypervisor host, network bandwidth etc it can take 2 to 3 hrs to complete.

- ▶ Once the `automated_install.sh` is complete logon to the bootstrap node. You need to be in the chef-bcpc directory on the hypervisor `vagrant ssh`
- ▶ Once logged onto the bootstrap node, cd to chef-bcpc directory
- ▶ Then run the following set of commands twice in sequence

```
./cluster-assign-roles.sh Test-Laptop hadoop bcpc-  
vm1
```

```
./cluster-assign-roles.sh Test-Laptop hadoop bcpc-  
vm2
```

```
./cluster-assign-roles.sh Test-Laptop hadoop bcpc-  
vm3
```

- ▶ This completes the creation of the three hadoop nodes bcpc-vm1 is a master node which hosts HDFS Namenode, HBase master, MySQL server and the ip is 10.0.100.11 bcpc-vm2 is a master node which hosts YARN resource manager and Hive/Hcatalog, MySQL Server and the ip is 10.0.100.12 bcpc-vm3 is the worker node which hosts HDFS Datanode, HBase region server, YARN node manager and the ip is 10.0.100.13

- ▶ System stats from the nodes and JMX stats from the various hadoop components are available through graphite. The URL to access is `https://10.0.100.5:8888`
- ▶ Monitoring of hadoop components are done through Zabbix and it can be accessed through the URL `https://10.0.100.5:7777`
- ▶ Passwords for various components including the password to login to the hadoop nodes can be retrieved by logging on to the bootstrap node and issuing the following command From chef-bcpc directory in hypervisor

```
vagrantssh
cd chef-bcpc
sudo knife data bag show configs Test-Laptop
```

This will list the user-id and password of all the components that are unencrypted. `sudo knife vault show os cobbler "root-password" --mode client` This will list the cobbler-root-password, which is stored in Chef Vault. This is the password to logon to the 3 hadoop nodes as the user "ubuntu" which is part of sudoers list.

Creating a kerberos principal : At this point we can start testing various components, since the cluster uses kerberos authentication we would need to create a kerberos principal for the ubuntu user.

```
root@bcpc-bootstrap:/home/vagrant/chef-bcpc# kadmin.local
Authenticating as principal root/admin@BCPC.EXAMPLE.COM with password.
kadmin.local: add_principalubuntu
WARNING: no policy specified for ubuntu@BCPC.EXAMPLE.COM; defaulting to no policy
Enter password for principal "ubuntu@BCPC.EXAMPLE.COM":
Re-enter password for principal "ubuntu@BCPC.EXAMPLE.COM":
Principal "ubuntu@BCPC.EXAMPLE.COM" created.
kadmin.local:
```

Giving ubuntu user hbase privileges : For the purposes of testing, make ubuntu a superuser

```
$ sudo -u hbasehbase shell
hbase(main):001:0> grant 'ubuntu', 'RWCA'
..skipping standard messages..
0 row(s) in 0.9490 seconds
The execute permission 'X' is unnecessary if users do not require hbase delegation tokens.
Hadoop Cluster Verification
```

Once cluster is created and setup, its health can be verified by examining the results of automated smoke tests or by running manual tests. Following sections explain the process to verify cluster using both automated and manual tests. The verification process requires a valid kerberos ticket which can be created by executing `kinit` statement. `kinit` statement would use the logged in user (ubuntu in this case) and would ask for password to complete the ticket creation process.

```
kinit
```

Automated Cluster Health Verification (Smoke Tests)

Chef-Bach also runs smoke tests to verify cluster health. These smoke tests are scheduled to run after every 10 minutes. To check cluster's health through smoke tests, please follow the steps below:

1. Oozie server is installed on vm2 node, make sure `OOZIE_URL` is set if you are running these steps from other nodes, i.e. vm3 or vm1 nodes.

```
export OOZIE_URL=http://f-bcpc-vm2:11000/oozie
```

1. Run following command to find out Job Id (Coordinator Id) for Oozie-Smoke-Test-Coordinator jobs.

```
oozie jobs --jobtype=coordinator
```

2. Use the Job Id from step 1 to find the workflow run.

```
oozie job -info <JobId_FROM_STEP 1>
```

3. Step 2 will list all the workflow runs, please copy the latest Ext ID (workflow ID). It will look like xxxxxxx-xxxxxxxxxxxx-oozie-oozie-W where x represents a random digit.

4. Workflow ID will show us result of all workflow actions by executing following command.

```
oozie job -info <WORK FLOW _ ID _ FROM _STEP3>
```

To make sure, cluster is healthy, all the tests should be succeeded.

Manual Cluster Health Verification

This section explains the process to verify cluster manually. Tests are performed on each Hadoop components to confirm its health.

HDFS

- ▶ Log on to bcpc-vm3. You can do ssh ubuntu@10.0.100.13 using the cobbler _root password from above from the hypervisor or from the bootstrap node chef-bcpc directory issue./nodessh.sh Test-Laptop 10.0.100.13 - Take note of the cobbler_root password that is printed if you used the second option to login. This is the password you will need for sudo below.
- ▶ Run `sudo -u hdfsdfsdfs -copyFromLocal /etc/passwd /passwd`
- ▶ Run `sudo -u hdfsdfsdfs -cat /passwd`
- ▶ Run `sudo -u hdfsdfsdfs -rm /passwd`
- ▶ If all these are successful the **hdfs** component is verified

HBase

- ▶ Run hbase shell
- ▶ Under the hbase shell, run `create 't1','cf1'`

- ▶ Run `list` which should display the newly created table as a list
- ▶ Run `put 't1','r1','cf1:c1','v1'`
- ▶ Run `scan 't1'` which should display the row create in the previous step
- ▶ Run `disable 't1'`
- ▶ Run `drop 't1'`
- ▶ Run `list` and it should display an empty list
- ▶ Run `exit`
- ▶ If all these steps are complete, the **HBase** component is verified along with ZooKeeper

Map/Reduce

- ▶ Assuming your username is xyz, we need to create that user on all the hadoop nodes and a home directory on HDFS
- ▶ Run the following on any node:

```
sudo -u hdfsdfsdfs -mkdir /user/xyz
```

```
sudo -u hdfsdfsdfs -chown xyz /user/xyz
```
- ▶ Create a new user in all the three hadoop nodes using `adduser` command
- ▶ Login to the bcpc-vm2 (10.0.100.12) node and switch to the new user created in the previous step.
- ▶ Run `yarn jar /usr/hdp/current/hadoop-mapreduce-client/hadoop-mapreduce-examples.jar pi 1 100` Note: Replace the hdp version based on your installation. You may need to make sure that the requested container size is within the yarn minimum container size (see yar-site.xml)

```
yarn jar /usr/hdp/current/hadoop-mapreduce-client/hadoop-map reduce-examples.jar pi -Dmapreduce.map.memory.mb=256 -Dmapreduce.reduce.memory.mb=512 -Dmap reduce.reduce.java.opts="-Xmx410m" -Dmap reduce.map.java.opts="-Xmx205m" -Dyarn. app. map reduce.am.resource.mb=256110
```
- ▶ If the previous step completes successfully it verifies **YARN** and **MapReduce** components.

FACULTIES OF COMPUTER SCIENCE

M.Sc. IV Semester Examination

Model Paper - I

INTERNET OF THINGS

Time : 3 Hours]

[Max. Marks : 80

PART - A (8 × 4 = 32 M)

Answer all the Questions

ANSWERS

1. Explain the following communication models of IoT
 - (i) Request-Response communication model
 - (ii) Publish-Subscribe communication model
2. Write about Smart Lighting.
3. What is M2M communication?
4. Write a note on SNMP.
5. What is Raspberry PI?
6. What is Amazon Simple Storage Service (Amazon S3) ?
7. Write a note on IoT printer.
8. What is YARN?

(Unit-I, Q.No. 4)

(Unit-I, Q.No. 8)

(Unit-II, Q.No. 1)

(Unit-II, Q.No. 7)

(Unit-III, Q.No. 2)

(Unit-III, Q.No. 16)

(Unit-IV, Q.No. 5)

(Unit-IV, Q.No. 11)

PART - B (4 × 12 = 48 M)

Answer any Four of the following questions

9. Explain about the layers of IoT architecture.
10. Explain about various IoT technologies used for smart cities.
11. What is Network Function Virtualization? Explain.
12. Explain about the steps involved in IOT design methodology.
13. What is Raspberry PI? Explain about the structure of a Raspberry Pi.
14. Explain how to design a RESTful Web API.
15. Write a Case study on Smart Parking.
16. What is a Hadoop Cluster? Explain.

(Unit-I, Q.No. 2)

(Unit-I, Q.No. 9)

(Unit-II, Q.No. 5)

(Unit-II, Q.No. 13)

(Unit-III, Q.No. 2)

(Unit-III, Q.No. 13)

(Unit-IV, Q.No. 2)

(Unit-IV, Q.No. 10)

FACULTIES OF COMPUTER SCIENCE**M.Sc. IV Semester Examination****Model Paper - II****INTERNET OF THINGS**

Time : 3 Hours]

[Max. Marks : 80

PART - A (8 × 4 = 32 M)*Answer all the Questions***ANSWERS**

1. Explain the following communication models of IoT
 - (i) Push-Pull communication model
 - (ii) Exclusive Pair communication model
2. Write about Smart Appliances
3. What are the features of M2M communication ?
4. Write a note on NETCONF protocol.
5. What is cloud storage model?
6. What is Amazon Relational Database Service (Amazon RDS)?
7. Write a note on the relationship between data analytics and IoT.
8. What is Chef?

(Unit-I, Q.No. 4)

(Unit-I, Q.No. 8)

(Unit-II, Q.No. 2)

(Unit-II, Q.No. 9)

(Unit-III, Q.No. 9)

(Unit-III, Q.No. 17)

(Unit-IV, Q.No. 6)

(Unit-IV, Q.No. 17)

PART - B (4 × 12 = 48 M)*Answer any Four of the following questions*

9. Explain the deployment levels of IoT.
10. Write about Smart Energy devices.
11. What is SDN? Write about the role of SDN in IOT.
12. Write about the case study on weather monitoring system using IoT.
13. Explain about Wireless Application messaging protocol for IoT.
14. Write about SKYNET IoT Messaging Platform.
15. Write a case study on smart agriculture.
16. What is Apache Spark? Explain about it.

(Unit-I, Q.No. 7)

(Unit-I, Q.No. 11)

(Unit-II, Q.No. 4)

(Unit-II, Q.No. 14)

(Unit-III, Q.No. 10)

(Unit-III, Q.No. 21)

(Unit-IV, Q.No. 4)

(Unit-IV, Q.No. 14)